MARKUP OF MATHEMATICAL FORMULAS

Z286
E43A76

Electronic Manuscript Project

ASSOCIATION OF AMERICAN PUBLISHERS
ELECTRONIC MANUSCRIPT SERIES

Markup of Mathematical Formulas

Version 2.0

November 1987

© Copyright 1987 Association of American Publishers

Other publications from the Association of American Publishers Electronic Manuscript Series:

Standard for Electronic Manuscript Preparation and Markup

Author's Guide to Electronic Manuscript Preparation and Markup

Reference Manual on Electronic Manuscript Preparation and Markup

Markup of Tabular Material

Available from: Association of American Publishers
2005 Massachusetts Avenue, NW.
Washington, DC 20036
Telephone: (202) 232–3335

ASSOCIATION OF AMERICAN PUBLISHERS

ELECTRONIC MANUSCRIPT SERIES

MARKUP OF MATHEMATICAL FORMULAS

An SGML Application Conforming to International Standard ISO 8879 — Standard Generalized Markup Language.

Version 2.0

Association of American Publishers, Inc.

2005 Massachusetts Avenue, N.W.
Washington, D.C. 20036
Telephone 202 232-3335

Z286
E43A76

ACKNOWLEDGEMENT

With grateful appreciation to the
Graphic Communications Association (GCA)
for providing members to the
Joint AAP/GCA Math Committee and
for cooperating in this effort.

© Copyright 1987 Association of American Publishers

CONTENTS

Introduction 7
Objectives of the AAP Math Markup 7
How to Use This Manual 8
Getting Started 8
Relationship Between Text and Mathematical Formulas 8
Specification of Type Styles 9
Spacing in Your Mathematical Formulas 9
Markup Minimization 9
Short Tags 9
1. Partial Short Tags 10
2. Empty Start- and End-tags 10
3. Null End-tags 11
Omission of Start-tags 11
Short References 11
Special Characters: Entity References 12
Using Entity References in Formulas 12
An Alternate Form of Entity References 12
The Basics of Marking Up Mathematical Formulas 12
Advanced Features 16
Formula 18
1. In-line Formulas 18
2. Displayed Formulas 18
Fraction 19
Limits 21
Radical 21
Fences 21
1. Parentheses, Square Brackets, and Braces 23
2. Vertical Bars 24
3. Left and Right Posts 25
4. Center Posts 26
Arrays 26
1. Column and Row Separators 27
2. Column Alignment 28
3. Array Cell Alignment 28
Stacks 29
Superiors and Inferiors 29
Boxed, Underlined, and Overlined Elements 30
Accented Elements 31
Character String Elements 33
1. Roman Functions 33
2. Increment 33
3. Generically-coded Symbols 33

Type Style Change Tags 34
Spacing and Positioning Tags 38
1. Turn Lines 38
2. Mark and Mark Reference 38
3. Horizontal and Vertical Spacing 39
4. Atom Change Tag 40
Phrases 40
Appendixes
Appendix A: Formal Syntax for Mathematical Formulas 45
Appendix B: Full Markup of Mathematical Examples 60
Appendix C: Entity References for Use in Mathematical Formulas 63
Appendix D: Comprehensive List of Mathematical Constructs with Tags 71
Appendix E: Description of Special Symbols in Table 1 72
Math Committee 80

MARKUP OF MATHEMATICAL FORMULAS

INTRODUCTION

The manuscript markup procedures described in this manual are an application of the Standard Generalized Markup Language (SGML) and follow the same rules as defined in the Association of American Publishers' (AAP) *Standard for Electronic Manuscript Preparation and Markup.* Markup rules for text are described in the AAP's Electronic Manuscript Series: *Author's Guide to Electronic Manuscript Preparation and Markup* and *Reference Manual on Electronic Manuscript Preparation and Markup.* This manual assumes that you are familiar with the information provided in those two documents. This AAP guide expands on the basic concepts of SGML/AAP markup. Therefore, you should first understand the fundamental rules for text markup before proceeding with this manual.

Objectives of the AAP Math Markup

Microcomputers, word processors, and electronic printers provide the basis for greatly improved communications and information exchange within the publishing community. The full potential of this technology, particularly for authors of scientific and technical documents, has not been realized for two reasons. First, device incompatibility exists. Second, there is a wide variety of coding notations currently being used for electronically created manuscripts.

These problems can be largely overcome with a generic markup approach that is device, system, and application independent. The Association of American Publishers has defined such a markup notation for electronically created manuscripts in its Electronic Manuscript Series. The "Author's Guide" and "Reference Manual" in the series include a description of the markup for text with special symbols and simple tables. This document builds on those basic manuals to describe rules for marking up mathematical formulas that will be embedded in text and for marking up stand-alone mathematical formulas.

This manual has three basic objectives. The first is to describe a markup scheme that can be used by technical and nontechnical professionals. Although mathematicians, scientists, and engineers will be the major users, they will not be the only users. Librarians, editors, typographers, and secretaries, among others, will also write, read, or edit files that incorporate this markup.

The second objective is that the mathematical formulas be input and marked up using only ISO 646 International Reference Version (IRV) characters. (The ASCII character set is the United States version of ISO 646 IRV.) Considering the large and rapidly growing number of devices used for manuscript input, it is essential that a manuscript transmitted to a receiving device be done in a common character set.

Since ISO 646 and ASCII are established standards in the computer and data transmission environments, they should be recognized by most devices involved in the electronic manuscript process.

The third objective is that the constructs used to build formulas follow formal definitions. In SGML and its AAP applications, the markup or tagging of document elements relies on the inherent structure of the document to keep markup to a minimum. Like the text of a document, the components of formulas have a well-defined organization. Thus, they can be formalized and the structure used to facilitate the markup of formulas. By formalizing the structure of the constructs, some of the placement of tags around and within the constructs can be automated, resulting in many abbreviated or omitted author-inserted tags. The shortening or omission of tags through application of predefined structure is called markup minimization. The specific techniques are described in the subsequent section called MARKUP MINIMIZATION.

How to Use This Manual

This manual is written on the assumption that you are only concerned with the proper identification of the elements in your mathematical formulas and not with the stylistic details (type sizes, fonts, spacing, and so on). As much as possible, the AAP mathematical markup approach avoids typographical technicalities and simply tells you how to key the formulas. Few users will need all the material in this manual, and some will need only a small part of it.

General information on the relationship between the text and formulas in your manuscript and some basics on what you do and do not have to be concerned with are provided in the section called GETTING STARTED.

The section on MARKUP MINIMIZATION is devoted almost entirely to additional techniques that are not described in either the "Author's Guide" or "Reference Manual." These special techniques will be employed in all examples throughout this manual.

Use of special characters or symbols is an important part of creating mathematical formulas. The section SPECIAL CHARACTERS: ENTITY REFERENCES describes how to use entity references to input mathematical symbols.

If you are interested simply in identifying the elements of your mathematical formulas so they can be properly displayed on paper or a video display screen or stored in and retrieved from a database, you need only read the section on THE BASICS OF MARKING UP MATHEMATICAL FORMULAS in addition to the first four sections. However, if you need features beyond the basics, you should also read the ADVANCED FEATURES section.

GETTING STARTED

Relationship Between Text and Mathematical Formulas

Mathematical formulas might be created and processed as the only content of a manuscript, or they might be interspersed with text in a manuscript. In the first case, the formal syntax given in Appendix A of this manual is all that is needed to mark up your manuscript. The formal syntax provides the document type declarations for mathematical formulas. When formulas are intermixed with text, it is necessary to incorporate the document type declarations for mathematical formulas into the document type definition of the basic, text-only document type you have chosen, such as book or article. The method for combining the document types is also described in Appendix A.

You need not be concerned with the details of the document type declarations nor with how to incorporate them; that will be the task of your publisher. What you do need to follow is the structure of the type of document you are creating (as outlined in the "Author's Guide" and "Reference Manual") and the structure of the mathematical constructs you are using in your formulas. The section titled THE BASICS OF MARKING UP MATHEMATICAL FORMULAS describes how to integrate text expressions directly into a formula.

Specification of Type Styles

Over the years it has become customary to display certain mathematical constructs or elements in particular styles of type, particularly when the manuscript is produced as a printed product. In general, mathematical symbols consisting of single lightface (not bold) letters are italicized ($E = mc^2$). However, abbreviations of units ($9.8\ \mathrm{m\ s^{-2}}$) and symbols consisting of two or more letters (cos; ln; Tr; Arg) are roman. Using the markup methods described in this document, you do not have to concern yourself with specifying the type styles for these or any other math constructs. Default values have been established and they are taken care of after you have created your manuscript, provided you key the proper generic codes.

The Greek, German, and Cyrillic characters are exceptions, however. The type style for characters in these alphabets must be indicated explicitly. Special tags are available for this purpose. They are explained in the Type Style Change Tags subsection under ADVANCED FEATURES below.

Some of the other elements or constructs that diverge from the customary defaults are contained in the subsection called Generically-coded Symbols. You are encouraged to consult that subsection to familiarize yourself with those elements. If you have other elements for which the type styles diverge from the defaults, you can specify the desired type style explicitly. Details are given in the ADVANCED FEATURES section, along with some typical examples.

Spacing in Your Mathematical Formulas

As with type style specification, you generally do not have to worry about the spacing in your formulas. For example, the spacing around mathematical symbols such as = or +, roman functions, differentials, etc., and between statements in a mathematical formula is normally taken into account by the processing system. If circumstances require that you explicitly specify the spacing in a particular math construct, you may do so with the horizontal and vertical spacing tags and the atom change tag described in the ADVANCED FEATURES section.

MARKUP MINIMIZATION

The "Author's Guide" and "Reference Manual" provide authors with two types of markup minimization: omission of end-tags and empty end-tags (< / >). (No spaces should precede or follow the slash in an end-tag or an empty end-tag.) Both forms are available to you in marking up your math constructs. Remember that the end-tag should only be omitted if doing so does not cause any ambiguities about which element is being ended and that the empty end-tag assumes the generic identifier of the most recently opened but unended element.

Short Tags

In addition to the availability of the empty end-tag, there are three types of short tags provided for use in marking up mathematical formulas. They are the partial short tag, the empty start-tag, and the null end-tag.

1. Partial Short Tags

A partial short tag is one in which part of the tag is omitted. There are two forms of partial short tags. The first omits the closing delimiter (>) of the first tag when it is followed immediately by another tag. For example:

```
<ar><arr>The first row in an array< / arr>< / ar>
```

can be abbreviated to this:

```
<ar<arr>The first row in an array< / arr< / ar>
```

The second form of partial short tag omits the attribute name and the equals sign in a start-tag. This is possible if a specific list of attribute values has been defined for use with a particular element. For example, the values for aligning the content in a specific array cell have been defined as (l)eft, (c)enter, or (r)ight. Therefore, this tag:

```
<arc align = r>
```

can be shortened to:

```
<arc r>
```

2. Empty Start- and End-tags

An empty start-tag is one in which the generic identifier (GI) and all attributes have been left out. The GI assumed is that of the next mandatory (i.e., required) element in the structure. Any attributes will assume the default values.

As an example of using an empty start-tag, assume that an array row is defined to include one or more cells. Using end-tag omission only, the markup would be the following:

```
<arr>
<arc>cell 1
<arc>cell 2
<arc>cell 3
< / arr>
```

With empty start-tag, the markup would be the following:

```
<arr>
<>cell 1
<arc>cell 2
<arc>cell 3
< / arr>
```

The GI of only the first cell element may be omitted because it is the only one that is required. The other two cell elements are optional and must, therefore, have an explicit start-tag.

Empty end-tags could have been used in place of the array row end-tag. Remember, however, that two empty end-tags (< / >< / >) would be required since the first one would end the cell element and the second would end the row element.

An empty end-tag can often be used to indicate the end of an element. The start-tag of the next element can then be omitted as long as each element in the construct is required and is allowed to occur only once in a specified sequence. As an example, assume that a fraction is defined to include a numerator followed by a

denominator. The markup, using end-tag omission only, would be the following:

```
<fr><nu>2x<de>3xy< / fr>
```

$$\frac{2x}{3xy}$$

By using partial short tags and empty end-tags, the markup could be shortened to the following:

```
<fr<nu>2x< / >3xy< / fr>
```

$$\frac{2x}{3xy}$$

This can be shortened even further as will be seen below.

3. Null End-tags

The third type of short tag is a null end-tag, which is the single character "/" (slash). The null end-tag is interpreted as an empty end-tag only for an element in which the slash was used in place of the start-tag's closing delimiter. For example:

```
x<sup>2< / sup>
```

x^2

can be shortened to:

```
x<sup / 2 /
```

x^2

Remember that when using the null end-tag the slashes must always appear in pairs. Otherwise, you will not get the results you want.

Omission of Start-tags

End-tag omission is used extensively in text manuscripts and is used in several of the examples given above. The end-tags can be omitted because the structure of the elements in a manuscript can be used to implicitly end other elements.

The same is true for start-tags. Since the constructs in mathematical equations have a well-defined structure that is formally defined, the start-tag for elements can sometimes be omitted. With the definition given above for a fraction and by omitting start-tags and using empty end-tag minimization, the markup can now be shortened to the following:

```
<fr>2x< / >3xy< / fr>
```

$$\frac{2x}{3xy}$$

All of these techniques will be employed in various combinations throughout this manual to illustrate the minimum amount of markup you have to use in your formulas. The full markup of each example given below will be included in Appendix B for comparison. You may use full markup instead of minimized markup if you prefer. The number to the left of each example matches the number of its corresponding fully marked up version in Appendix B.

Short References

The short references defined for use in the regular text of your manuscript may also be used in the phrase elements of your formulas. Additional short references have been defined for a few of the diacritical marks and for use in the fence, array, and accented element constructs. They are discussed in subsequent sections.

You are encouraged to also use short references for any frequently used entity references or tags if you wish. Details for defining short references can be found in Appendix E of the "Reference Manual."

SPECIAL CHARACTERS: ENTITY REFERENCES

Using Entity References in Formulas

An entity reference is a string of characters from the ISO 646/ASCII character set that is used to refer to another character, a symbol, a file located outside the manuscript, an abbreviation, etc. Since mathematical formulas frequently include characters or symbols that are not part of the ISO 646/ASCII character set, entity references are used to reference those characters and symbols.

A number of entity references for applications other than mathematical formulas are listed in Appendix C of the "Reference Manual." In addition, entity references for mathematical symbols are listed in Appendix C of this manual.

An Alternative Form of Entity References

In order to simplify the process of locating the symbols given in Appendix C, they have been placed in Table 1. The coordinates of the table may be used as the entity references instead of those given in Appendix C. When using the table, the sequence of characters for the entity reference is the top coordinate, followed by the side coordinate. As an example, the "for all" symbol (top row, fourth column) will have the following entity reference if the table coordinates are used—`&ja;`.

The table is organized so that all arrows are located in the `&a` columns, most "verbs" are located in the `&v` columns, and all negated symbols are in the `&n` columns. No specific organization is used for the other columns.

THE BASICS OF MARKING UP MATHEMATICAL FORMULAS

This section provides a list of the constructs, in their simplest form, that you may use in creating a mathematical formula. For more details about the constructs and their respective attributes that are available to you, refer to the ADVANCED FEATURES section.

Formula

To Get	*Key*	*Comments*
In-line formula	`<f>rest of formula</f>`	A formula that is located within the text.
Displayed formula	`<fd>rest of formula</fd>`	A formula that is set off from the surrounding text. It consists of two parts—an optional label and at least one formula line. The label is shown below and the tag for the first formula line does not have to be keyed.
Displayed formula with label	`<fd<la/label/rest of formula</fd>`	

TABLE 1. Special symbols

	&a	&v	&n	&j	&k	&l	&m	&p
a;		⊑	⊈	∀	∧	ℵ	–	
b;	↓	⊂	⊊	∃	⊕	ב	♀	
c;	↓	⊂	⊄	⊳	∈		C	
d;	↓	⊒	⊉	)>	▵	ℸ		
e;	⇓	⊇	⊋	∞	ⓘ	∋	∈	
f;	⇋	⊃	⊅	⩽	⋹	⩽	♀	
g;	⇌	⋙	≯	$>^3$	>	≫	♂	
h;	⇆	⧧	≯	ℏ		ב	♄	
i;	⇄	∝	≯	$\int^2$	∫	∮	∮	
j;	↩	⊩	⊁	≍	∩	∯	∮	
k;	←	⊫	⊁	≥		∰	♮	
l;	←	⋘	⇍	$<^3$	<	≪		
m;	⇐	⊩	⇏	∓	⋉	♭	♂	
n;	↔	≤	≰	≠	⋊	♯		
o;	⇝	≥	≱	⩾	⋈	⩾	∅	
p;	↪	⊥	∄	∥	∦	⊥	℘	
q;	→	∥	∦	∨	⩛		∵	
r;	→	≙	↛	—∘	⋔		∴	◧
s;	⇒	≑	⇏		⊼	■	□	◹
t;	↦	≏	↛	~	⊤	⌙	⊤	◩
u;	↑	≎	≠		⋓	$\uplus^1$	$\cup^1$	◺
v;	⇑	∣	∤	∣	∤		$\vee^1$	◪
w;	↑	≦	≨		⋒		$\cap^1$	
x;	↑	≧	≩	×	≺	⊠	$\otimes^1$	
y;	⇛	≤	≨	⁕	≻		$\wedge^1$	≳
z;	⇐	≥	≩			⊡	$\odot^1$	≲

	&a	&v	&n	&j	&k	&l	&m	&p
0;	↕	≡	⊭	°	⬠	●	○	◒
1;	⇕	≃	≠		⬡	◆	◇	◆
2;	↔	≅	⊯	⇎	▽	♥	◇	
3;	⇔	≀	⇔		△	◦	Ψ	
4;	↖	≲	≰	‡		♭	∠	
5;	↘	≳	≱	†	ıı	♠	∡	
6;	↗	◁	⋭	`	◁	♣	∢	
7;	↙	▷	∉	′	▷	$\amalg^1$	⁞	
8;	⇈	⋚	≮	″	⋪	⋚	≤	
9;	⇊	⋛	≯	‴	⋫	⋛	≥	
10;	⇉	≃	≮	·	≙	▲	△	
11;	⇇	≳	≯		≐	▼	▽	
12;	◡	≋	≭	✠	≕	◀	◁	
13;	◠	≥	≯	÷	≔	▶	▷	
14;	∩	≺	≭	◡	◡	$\sqcup^1$	⊻	
15;	∩	≻	≱	◠	◠	⊓	⊼	
16;	⌐	≼	⊈	‰	‰		⊙	
17;	¬	≳	≱	≒	≒		⊙	
18;		≅	≠		÷	⊟	⊖	¬
19;		⊏	⊁	⟨	≵		∅	
20;		⊐	⊀	⟩	≵			
21;	↣	⊑	≺		☆		★[1]	
22;	↢	⊒	⊀		≡	≡	≀	
23;	⇔	≃	⊬					
24;	←	≼	⊈		⊣			
25;	→	≽	⊉		⊢	+		
26;	↢	⊢	⊬	±	+	⊞	$\oplus^1$	⊣
27;	↣	⊨	⊭		⊥			⁄
28;	↣	≻	≠	≡	=	═	≈	
29;	↰	≽						
38;	↱	≅	≠		/	\		\
31;				*	*	⊛		

	&a	&v	&n	&j	&k	&l	&m	&p

NOTES:

1. If this character is desired as a cumulator, include it as the operator in the limits construct. (No operator is required for the special limits constructs: summation, product, and integral.)
2. This character should only be used in the limits construct if multiples are desired. Otherwise, use the special limits construct for integral.
3. Do not use this character as an angle bracket. (See `&j19;` and `&j20;`)

Fraction

To Get	Key	Comments
numerator / denominator	`<fr>numerator< / > denominator< / fr>`	The fraction bar is inserted automatically and the narrower of the numerator or denominator will be centered with respect to the fraction bar.

Limits

To Get	Key	Comments
upper operator operand lower	`<lim>operator< / > lower< / >upper< / > operand< / lim>`	The operand is an optional element in all limits constructs.
$\sum_{\text{lower}}^{\text{upper}}$ operand	`<sum>lower< / >upper < / >operand< / sum>`	
$\int_{\text{lower}}^{\text{upper}}$ operand	`<in>lower< / >upper < / >operand< / in>`	
$\prod_{\text{lower}}^{\text{upper}}$ operand	`<pr>lower< / >upper < / >operand< / pr>`	

Fences

To Get	Key	Comments
|content|	`<fen>content <rp>< / fen>`	The parentheses, square brackets, and curly brackets can be used only as short references for fences within mathematical formulas.
(content)	`(content)`	
[content]	`[content]`	
{content}	`{content}`	
|content|content|	`<fen>content<cp> content<rp>< / fen>`	

Arrays

To Get	Key	Comments
$A \quad B$ / $C \quad D$	`<ar>@A|B @C|D< / ar>`	The content of each cell will be centered within the cell by default. The `@` and `|` are short references for `<arr><arc>` and `<arc>`, respectively, in an array.
$\begin{pmatrix} A & B \\ C & D \end{pmatrix}$	`(<ar>@A|B @C|D< / ar>)`	

Stacks

To Get	Key	Comments
layer / l a y e r / layer	`<stk>layer<lyr>l a y e r <lyr>layer< / stk>`	The default alignment of each layer will be centered with respect to the layer above.

Radical

To Get	Key	Comments
$\sqrt[3]{a+b}$	`<rad>a + b<rdx>3< / rad>`	The index element is optional. The radical sign will be generated automatically.

Superiors and Inferiors

To Get	Key	Comments
x^{i}	`x<sup / i /`	
x_{i}	`x<inf / i /`	
x_{n}^{i}	`x<sup / i / <inf / n /`	The superior should be keyed first when you have a sequence of superior and inferior.
$x^{i''}$	`x<sup / i<sup / n / /`	
x^{i_n}	`x<sup / i<inf / n / /`	

Accented Elements

To Get	Key	Comments
ā	`<a / a|&d1;/`	The vertical bar is a short reference for `<ac>`, an accented element component.
ã	`<a / a|~ /`	

Type Style Tags

To Get	Key	Comments
β	`<g / b /`	See Table 6 for the appropriate characters to use as the content of the mathematical Greek tag (`<g>`).
∂	`<g / 6 /`	
a	`<b / a /`	

Phrases in Formulas

To Get	*Key*
force = mass × acceleration	`<phr / force / = <phr / mass / &jx; <phr / acceleration /`

Character String Elements

To Get	*Key*
Vector; e.g., **v** or $\vec{v}$	`<v / v /`
Increment; e.g., δu	`<inc / <g / d / u /`
Roman function; e.g., sin	`<rf / sin /`

ADVANCED FEATURES

This section provides you with the details, as well as the advanced features, for marking up your mathematical formulas. The advanced features include attributes that may be used with certain constructs, tags for specifying type styles other than the defaults, and a number of spacing and positioning tags. All of the advanced features described in this section are optional and need not be used in your mathematical formulas unless you need them.

The elements making up a construct and the number of times they may occur are given for each construct. In addition, the markup for at least one example is given for each construct.

Note that constructs are themselves elements, usually composed of other elements. Throughout this manual a complete unit will usually be referred to as a construct and its components will be called elements. Note also that formulas are often built up by nesting the described constructs. You may nest any construct within another as long as it is permitted to occur in the containing construct. If the nesting becomes very deep, the full markup as shown in Appendix B may be easier to use.

The contents of the constructs and the elements within the constructs have been divided into distinct groups according to type and complexity. The constructs and elements, along with their tags, for each group are:

Character strings group

— Accented element (**<a>**)
— Type styles
- Bold (**<b>**)*
- Bold German (**<bge>**)*
- Bold Greek (**<bg>**)*
- Bold italic (**<bi>**)*
- Bold italic sans serif (**<bisf>**)*
- Bold sans serif (**<bsf>**)*
- Bold script (**<bsc>**)*
- Greek (**<g>**)*
- Italic (**<it>**)*
- Italic sans serif (**<isf>**)*

- Monospace ("typewriter") (**<ty>**)*
- Openface ("blackboard boldface") (**<op>**)*
- Roman (**<rm>**)*
- Sans serif (**<ssf>**)*
- Script ("fancy") (**<sc>**)*
- Small caps (**<scp>**)*

*End-tag required (the empty end-tag may be used).

Character string text group

— Character data (text)
— Any element from the Character strings group

Special characters / strings group

— Roman function (**<rf>**)
— Increment (**<inc>**)
— Vector (**<v>**)
— Dyadic (**<dy>**)
— Field (**<fi>**)

Built-up elements group

— Fraction (**<fr>**)
— Radical (**<rad>**)
— Limits (**<lim>**)
— Summation limits (**<sum>**)
— Product limits (**<pr>**)
— Integral limits (**<in>**)
— Array (**<ar>**)
— Fence (**<fen>**)

Horizontally-oriented elements group

— Box element (**<box>**)
— Underline element (**<unl>**)
— Overline element (**<ovl>**)
— Superior (**<sup>**)
— Inferior (**<inf>**)

Complex built-up constructs group

— Any element from the Character string text group
— Any element from the Special characters/strings group
— Any element from the Built-up elements group
— Any element from the Horizontally-oriented elements group

Nonmath text group

— Paragraph (**<p>**)
— Quotation, in-line (**<q>**) (end-tag required)
— Type styles
- Bold (**<b>**)*
- Bold italic (**<bi>**)*
- Italic (**<it>**)*
- Roman (**<rm>**)*
- Small caps (**<scp>**)*

*End-tag required (the empty end-tag may be used).

— Lists
- List (<l>)*
- List, type 1 (<l1>)*
- List, type 2 (<l2>)*
- List, type 3 (<l3>)*
- List, type 4 (<l4>)*
- List, type 5 (<l5>)*

*End-tag required.

— References
- Appendix, reference to (<apr>)*
- Artwork, reference to (<artr>)*
- Bibliographic reference, reference to (<rb>)*
- Figure, reference to (<fgr>)*
- Footnote, reference to (<fnr>)*
- Note-in-text, reference to (<ntr>)*
- Source note, reference to (<srr>)*
- Table, reference to (<tbr>)*
- Label, reference to (<lar>)*

*End-tag required (the empty end-tag may be used).

— Character data

When an element or construct described below is composed of the elements in one of the above groups, the appropriate group will be referenced rather than repeating the elements in the group each time.

A complete list of the math constructs and their embedded elements, along with the tag for each, is shown in Appendix D of this manual.

Formula

A formula is a passage of mathematics. The passage may be one of two types: in-line, meaning located within the text, or displayed, meaning set off from the surrounding text.

1. In-line Formulas

The tag for an in-line formula is the following:

```
<f>
```

Any of the following constructs, except for the displayed formula, may be included within an in-line formula. The end-tag for the in-line formula (<f>) is required.

2. Displayed Formulas

The tag for a displayed formula is the following:

```
<fd>
```

A displayed formula is composed of two labels, both optional, and one or more formula lines. The end-tag for a displayed formula (<fd>) is required.

A label is similar to giving the formula a name. A formula may have zero, one, or two labels. If you don't care whether the label is placed on the left or right of the formula, use the following tag:

```
<la>
```


If your formula has only one label and you want to specify its placement, use the appropriate tag from the following list:

left side:	<la pre>
right side:	<la post>

Occasionally, it is necessary to give a formula two labels. In those cases both of the above tags must be used. A label is composed of any element in the Character string text group and the Horizontally-oriented elements group. If every line in the displayed formula is to be labeled, the lines must be input as individual displayed formulas with a label for each one. The label end-tag is required.

The label for any formula may be referenced from the text of the manuscript or from the nonmath text inside the formula. The technique for referencing the label will be described in the following subsection titled Phrases.

Formula lines are to be used for multiple-lined displayed formulas when you wish to specify the explicit line breaks in the formula. The first formula line does not require a tag since it is expected to be there and so is assumed. All subsequent formula lines must have a start-tag but do not have to have an end-tag. The start tag for a formula line is the following:

```
<fl>
```

A formula line is composed of one or more occurrences of any elements in the Complex built-up constructs group. The elements from that group may be repeated and may occur in any order.

To summarize this subsection, you have four choices for beginning your displayed formula. Examples of those choices follow:

No. of labels		*First formula line (start-tag omitted)*
no label:		<fd>
one label with unspecified location:	[1]	<fd<la/1.1/
one label with specified location:	[2]	<fd<la pre/1.1/
or:	[3]	<fd<la post/1.1/
two labels:	[4]	<fd<la pre/1.1.a/<la post/1.1.b/

The remainder of a formula line will be one or more of the constructs described below.

Fraction

A fraction is composed of a numerator followed by a denominator. Since both are required to be in a fraction and the numerator is always first, the start-tag for the numerator may be omitted and an empty start-tag for the denominator may be used. The form of a fraction is the following:

```
<fr>numerator</>denominator</fr>
```

The end-tag for the fraction is required. The contents of the numerator and denominator may be one or more of the elements in the Complex built-up constructs group.

Several simple examples of a fraction follow:

[5] `<fr>1</>2</ fr>` $\frac{1}{2}$

[6] `<fr>2x</ >3xy</ fr>` $\frac{2x}{3xy}$

[7] `<fr>2x + 3y</ >12x&ma;12y</ fr>` $\frac{2x + 3y}{12x - 12y}$

The fraction bar will be inserted automatically and the narrower of the numerator or denominator will be centered with respect to the fraction bar.

A fraction may be specified as a case fraction, a built-up fraction, or one using a slashed or solidus fraction bar. An example of each follows:

Shape	*Example*
case	$\frac{1}{2}$
built-up	$\frac{a+b}{c+dx}$
solidus	1/2

If you want to specify one of these shapes for your fraction, you may do so by using an attribute called shape in the fraction start-tag. The appropriate tags are as follows:

Shape	*Tag*
case	`<fr case>`
built-up	`<fr built>`
solidus	`<fr sol>`

(Caution: Be sure to use the fraction tag and structure to key a solidus fraction. Do not key it as 1/2, for example.)

The fraction bar is normally as long as the wider of the numerator or denominator. The narrower of the two is then centered with respect to the fraction bar. If you have a requirement to change that alignment, you may do so with the align attribute. The permitted values for align are "r" for right and "l" for left. Following is an example of an alignment to the right:

[8] `<fr r>a + b + c</ >x + y</ fr>` $\frac{a+b+c}{x+y}$

If you wish to change the style of the fraction bar, you may do so with the style attribute. The permitted values are shown in Table 2. The default style is "s," so style need only be specified if you want a style other than single. An example of a fraction with a bold fraction bar follows:

[9] `<fr b>1</ >1 + <fr>1</ >y</ fr</ fr>` $\frac{1}{1+\frac{1}{y}}$

You are permitted to use more than one attribute in the fraction start-tag if necessary. Some examples include:

Tag	*Description*
`<fr b sol>`	bold fraction bar, solidus shape
`<fr dot case>`	dotted fraction bar, case shape
`<fr l da>`	left alignment, dashed fraction bar

If the numerator or denominator is very long and you wish to indicate where a line break could occur, you may do so with the turn line tag described later in this section.

Limits

The limits construct is composed of a main symbol—called the operator, a lower limit, an upper limit, and the operand. The operand is that portion of the limits routine that is affected by the operator. Although you are not required to do so, you are encouraged to tag the operand so it will be recognized as a single element to ensure that it is treated properly.

The operator may be composed of an element from the Character string text group, the Horizontally-oriented elements group, or a roman function. The lower limit, upper limit, and operand may consist of one or more elements from the Complex built-up constructs group.

The markup for the limits construct is the following:

`<lim>operator</ >lower limit</ >upper limit</ >operand</ lim>`

For example,

$$\cup_{n=1}^{m}(x_n \cup y_n)$$

would be marked up as follows:

[10] `<lim>μ</ >n = 1</ >m</ >`

`(x<inf / n / μy<inf / n / )</ lim>`

(Note: if the operator is "lim," refer to the Roman Functions subsection for a description of how to mark it up.)

There are three operators which are used much more frequently than any others. They are the summation, product, and plain integral signs. Because of their frequency of use, they are defined as special limits in this manual. They differ from the general limits construct described above in that the operator will be inserted for you automatically. This is accomplished by indicating the type of limits with a special tag rather than the general limits tag. For example, the tag **<sum>** is used for a summation limits construct.

The tags for the three special limits constructs are the following:

summation limits:	`<sum>`
product limits:	`<pr>`
plain integral limits:	`<in>`

The result of using one of these special tags is that the construct does not need an

operator and is defined as a lower limit followed by an upper limit followed by an operand. The markup for each is as follows:

summation limits:

```
<sum>lower limit</>upper limit</>operand</sum>
```

product limits:

```
<pr>lower limit</>upper limit</>operand</pr>
```

plain integral limits:

```
<in>lower limit</>upper limit</>operand</in>
```

Following is an example of a summation limits construct and its markup:

[11] `<sum>i = 1</>i = 10</>x<inf/i/</sum>` $\sum_{i=1}^{i=10} x_i$

(The markup for the inferior will be described later.)

The placement of the limits with respect to the main symbol (i.e., centered above and below or to the right) is normally specified by your publisher. If you want to specify their placement yourself, however, you may do so with the align attribute. The permitted values for align in a limits tag or one of the special limits tags are "c" for centered and "r" for right. Some example tags include the following:

Tag	*Description*	*Example*
`<lim r>`	limits placed to the right of the main symbol.	$\bigcup_{n=1}^{m}(x_n \cup y_n)$
`<sum c>`	limits centered above and/or below the summation symbol.	$\sum_{i=1}^{i=10} x_i$

Radical

A radical is composed of a radicand and an optional index. If the index is included, the start-tag (`<rdx>`) must be used. The radical symbol will be inserted for you automatically. The radicand and the index may be composed of any elements included in the Complex built-up constructs group.

As an example of marking up this construct, the following radical

$\sqrt[3]{a+b}$

would have this markup:

[12] `<rad>a + b<rdx>3</rad>`

The end-tag is required for the radical.

Fences

Characters such as

( ) [ ] { } | ‖ ⌈ ⌉ ⌊ ⌋ ⟨ ⟩ ⌈ ⌉ ⌊ ⌋

that are used to enclose or set off other characters, are collectively called fences. They are considered a construct in this manual because they must often change size to match the constructs to which they relate. In order for the sizing to be done for you automatically, it is necessary to indicate their function as that of being fences.

A fence construct may be structured in either of two ways:

1. It may be composed of a left post, followed by any elements—repeatable and in any combination—from the Complex built-up constructs group, followed by a right post (`<rp>`). The general structure is the following:

```
<fen>contents<rp></fen>
```

2. Or, it may be composed of a left post, followed by any elements—repeatable and in any combination—from the Complex built-up constructs group, one or more center posts (`<cp>`) each followed by any elements—repeatable and in any combination—from the Complex built-up constructs group, and finally a right post. The general structure is the following:

```
<fen>contents<cp>contents<rp></fen>
```

The left post is always identified by an attribute, called lp (for "left post"), in the fence start-tag. By using markup minimization, the "lp = " does not have to be included in the tag, however. An attribute called post has been defined in the right post tag (`<rp>`) for indicating the desired right post. The default value for lp in the fence start-tag and post in the right post tag is "vb" (vertical bar). Additional attribute values available to you for posts are the following:

Value	*Meaning if value of lp*	*Meaning if value of post*
par	(	)
sqb	[	]
cub	{	}
ceil	⌈	⌉
fl	⌊	⌋
ang	⟨	⟩
sol	/	/
vb	|	|
uc	⌜	⌝
dc	⌞	⌟
llsqb*	[	[
rrsqb*	]	]

*If "llsqb" or "rrsqb" is used, it must be the value for both lp and post.

In order to simplify further explanation of how to mark up fence constructs, we will consider several types of fences.

1. Parentheses, Square Brackets, and Braces

The characters ( ), [ ], and { } are included in the ISO 646/ASCII character set and are treated as ordinary characters in the textual portions of your manuscript. In mathematical contexts, however, they are more than just ordinary characters; they have special meaning. Because these characters are used in mathematical formulas

more frequently as posts than as textual characters, they have been defined as short references when inside a formula. The short references are as follows:

In place of	*Use*
For left posts:	
`<fen sqb>`	[
`<fen cub>`	{
`<fen par>`	(
For right posts:	
`<rp sqb</fen>`	]
`<rp cub</fen>`	}
`<rp par</fen>`	)

These characters may be used only as fence short references when inside a formula. If you have need of any of these characters as text characters in a formula, you must input them as entity references. (See Appendix C in the "Reference Manual.")

Some examples follow:

For

$$(0,1)$$

the markup is

[13] `(0,1)`

For

$$\left[\frac{A+1}{B}\right]$$

the markup is

[14] `[<fr>A+1</>B</fr>]`

2. Vertical Bars

Like the post characters discussed above, the vertical bar is an ISO 646/ASCII character that is considered ordinary text in the textual portion of your manuscript but has a special meaning in a formula. As mentioned above, it is defined as the default value of the lp attribute in the fence tag and of the post attribute in the right post tag. Therefore, the fence construct

$$\left|\frac{A+1}{B}\right|$$

may be marked up as follows:

[15] `<fen<fr>A+1</>B</fr><rp></fen>`

If desired, you can specify different styles for the vertical bar post. This is possible with the use of another attribute called style. The allowed values for the style attribute are shown in Table 2. Modifications of the above example follow to illustrate the use of the style attribute.

For

$$\left|\frac{A+1}{B}\right|$$

you may key

[16] `<fen b><fr>A+1</>B</fr><rp b></fen>`

For

$$\left\|\frac{A+1}{B}\right\|$$

you may key

[17] `<fen d><fr>A+1</>B</fr><rp d></fen>`

Note that when an attribute is used, an empty start-tag may not be used nor may the closing delimiter be omitted.

TABLE 2. Style Attributes

Value	*Meaning*	*Example*
s	single	|
d	double	‖
t	triple	⦀
da	dashed	¦
dot	dotted	⋮
b	bold	┃
bl	blank	space between
n	none	no space between

3. Left and Right Posts

A fence construct is defined to include a left post and a right post. All the examples shown so far have included both posts. There are occasions, however, when either the left post or the right post is missing. In order to indicate a missing post in your fence construct, simply give the style attribute the value "n" for none (i.e., `<fen n>` or `<rp n>`). The post that isn't missing may be input by using any of the appropriate techniques discussed so far. For instance, this example

$$\frac{df}{dx}(y)\Big|_{x=0}$$

may be marked up as follows:

[18] `<fen n><fr><inc/df/</><inc/dx/</fr>`
`(y)<rp></fen><inf/x=0/`

Other circumstances might require that the right and left posts be different characters. You can assign almost any of the permitted values to the lp attribute and any other one to the post attribute. The exceptions are if either of the attribute values "llsqb" or "rrsqb" is being used, as noted in the chart above. Following is an example of differing left and right posts:

$$(0,1] \quad \text{keyed as} \quad \texttt{(0, 1]}$$

4. Center Posts

Some fence constructs include one or more center posts in addition to the left and right posts. The tag for a center post is **<cp>**. A center post has the same two attributes as a right post, post and style. Their default values are "vb" (vertical bar) and "s" (single), respectively. They may be assigned different values in the same way the right post attributes are changed. An example follows:

$$\left\langle \alpha + \beta \middle| \frac{\partial \alpha}{\partial p} \right\rangle$$

may be keyed as

[19]

```
<fen ang><g/a/+<g/b/<cp><fr><inc/<g/6/<g/a//</>

<inc/<g/6/p/</fr><rp ang></fen>
```

Arrays

Arrays are rectangular arrangements of elements in rows and columns. The tag for an array is **<ar>**. An array may consist of one or more rows, and the rows are composed of one or more cells. Their respective tags are **<arr>** and **<arc>**.

The contents of the array are contained in the cells. Any of the elements in the Complex built-up constructs group—repeatable and in any order—may be included in any array cell. The input sequence for an array is to start at the top row, leftmost cell, and proceed from left to right and top to bottom until the bottom row, rightmost cell is reached. Given the following array

$$\begin{pmatrix} 1a & 1b & 1c \\ 2a & 2b & 2c \end{pmatrix}$$

the markup might be the following:

[20]

```
(<ar<arr>1a<arc>1b<arc>1c
<arr>2a<arc>2b<arc>2c</ar>)
```

Note that 1) the start-tag for the first array cell in each row (**<arc>**) may be omitted; and 2) the array end-tag is required.

The markup for an array may be minimized even more by using short references. Therefore, the following short references may be used in an array, if desired:

In place of	*You may use*
<arr<arc>	@
<arc>	|

The markup for the above example would then look like this:

```
(<ar>@1a|1b|1c
@2a|2b|2c</ar>)
```

Array cells may have no content. If the empty cells occur at the end of the row, their tags may be omitted; otherwise, simply key the start-tag or short reference for the array row/cell followed immediately by the start-tag or short reference for the second cell and any subsequent empty cells. As an example:

$$\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$$

key the following:

[21]

```
(<ar>@1@|1@||1</ar>)
```

Each row may be input as a separate line if you wish. For instance, the above example could be input as follows:

[22]

```
(<ar>

@1

@|1

@||1

</ar>)
```

The end result will be the same in both cases.

1. Column and Row Separators

Occasionally, you might need to indicate that an array has a border and/or column and row separators. This may be done with attributes in the array start-tag. Vertical rules may be specified with the cs (column separator) attribute. The columns are identified with a "number" from left to right, with 1 being the leftmost column and F being the rightmost or final column. The column separators are labeled 0 through F, with 0 being to the left of column 1 and 1 through F being to the right of their corresponding columns. The styles for the rules are those shown in Table 2. The value of the cs attribute is a list of pairs. The first item in each pair is the column separator number, and the second item is the desired style of separator. The default style for all the column separators is "bl" (blank). Any unspecified column separators will remain blank. Examples:

<ar cs="0 d F d">	gives double rules to the left and right and blank separators between the columns.
<ar cs="0 d 1 b F d">	gives double rules to the left and right, a bold rule to the right of column 1 and blank separators everywhere else.

If you want to change all of the separators to something other than blanks, you may use "A" for all or all others (including to the left of column 1 and the right of

column F). The following tag would change all column separators to single rules:

```
<ar cs="A s">
```

Horizontal rules may be indicated with the rs (row separator) attribute. The technique is the same as that used for column separators, with 0 being above row 1 and F being below the final or last row. The number "A" applies to all or all other row separators (including above row 1 and below row F). Examples:

`<ar rs="0 d F d">` gives double rules above and below the array and blanks between all the rows.

`<ar rs="0 d 1 b F d">` gives double rules above and below the array, a bold rule below row 1, and blanks between all the other rows.

The following is an example of a start-tag for an array with a double rule border and single rules between all rows and columns:

[23] `<ar cs="0 d F d A s" rs="0 d F d A s">`

1	2	3
4	5	6
7	8	9

Note that the "`cs=`" and "`rs=`" must be included in the tag and that the value of each must be enclosed in quotes.

2. Column Alignment

The alignment of the contents of the columns in an array are centered within the column by default. If you wish to change the alignment within one or more columns, however, you may do so with the ca (column align) attribute. As with column separators, the columns are "numbered" from 1 through F. The permitted values for the alignment are "l" for left, "r" for right, "e" for exponent, and "d" for decimal point. (Centered alignment, "c", need not be specified since it is the default.) As with cs and rs, the value of ca is a list of pairs, the first item in each pair being the column number and the second the desired alignment. Again, the "`ca=`" must be included in the tag and the value must be in quotes. "A" may be used for all or all other alignments. An example follows:

[24]
```
1  2.1
2  3.64
```
`<ar ca="2 d">@1|2.1@2|3.64</ar>`

Note that you may use all three attributes, cs, rs, and ca, in a single array tag.

3. Array Cell Alignment

For situations when an individual cell requires an alignment other than that specified for the column (whether the default or specified), an attribute called align has been defined for use within the array cell start-tag. Its permitted values are the same as for column alignment (i.e., l, r, c, e, and d). When this attribute is used, the specified value is in effect for that cell only. An example of a single array row with

the alignment of one cell changed from the default to left follows:

[25]
```
492  362  5%
```
`@492|362<arc l>5%`

Note that the short reference may not be used if you need to include an attribute in the tag.

Stacks

A stack is a construct in which one thing is "stacked" on top of another. It is composed of one or more layers. The width of the stack is determined by the longest layer. The start-tags for stack and layer are `<stk>` and `<lyr>`. A layer may contain any of the elements in the Complex built-up constructs group—repeated and in any order. An example of a stack and the markup for it follow:

a
ABCBA
iωt

[26]
```
<stk>a
<lyr>ABCBA
<lyr>i<g/w/t</stk>
```

The default alignment of each layer is centered with respect to the layer immediately above it. The alignment of a layer may be changed to be flush left (l) or flush right (r) relative to the layer above by using an attribute called align. For example, to get

a
ABCBA
iωt

you would key

[27] `<stk>a<lyr l>ABCBA<lyr r>i<g/w/t</stk>`

Superiors and Inferiors

Superiors and inferiors, also called superscripts and subscripts, may consist of any elements in the Complex built-up constructs group. The start-tags for superior and inferior are `<sup>` and `<inf>`.

A superior or an inferior applies to the preceding character or element, and may only follow a character or an end-tag. Some examples of marking up superiors and inferiors follow:

[28] T_2^1 `T<sup/1/<inf/2/`

[29] e^{x^2} `e<sup/x<sup/2//`

[30] 2^{n_1} `2<sup/n<inf/1//`

[31] E_{a^3} `E<inf/<g/a/<sup/3//`

[32] $e^{a_2^1}$ `e<sup/a<sup/1/<inf/2//`

Note that if a symbol has both a superior and an inferior, the superior should be keyed first.

If you desire a staggered arrangement for the superiors and inferiors, you may indicate that desire by separating them with an element called a zero-width character (**<zw>**). A zero-width character has no content; its only purpose is to serve as an explicit separator. Following is an example of its use:

[33] $T_1^{2}{}_3$ `T<inf / 1 / <zw><sup / 2 / <zw><inf / 3 /`

There is a small category of characters that are positioned like superiors; but they aren't actually superiors because their normal position is raised off the baseline. They include the following:

Symbol	*Name*	*Keyed*
*	asterisk	*
°	degree sign	&j0;
′	prime	&j7;
″	double prime	&j8;
‴	triple prime	&j9;
‵	reverse or backwards prime	&j6;

These characters should be keyed as ordinary characters or entity references, and not as superiors. Their position will be the same as any other superior. Therefore, if you do not want them placed over an adjacent inferior, you must separate them with a zero-width character tag.

The asterisk occasionally appears in a lowered position like an inferior. It should also be keyed using only the entity reference **&j31;** (down asterisk), rather than as an inferior asterisk.

Most superiors and inferiors appear to the right of their subject. There are times when they appear to the left of their subject, however. An attribute called loc (for location) has been defined for inferiors and superiors to indicate placement to the left. The value "pre" will indicate that the superior or inferior should appear on the left of its subject. An example follows:

[34]

```
(<fr>a< / >b< / fr>)<sup pre / 1 / <inf pre / 2 /
<sup / cd / <inf / ab /
```

$$\left(\frac{a}{b}\right)_{ab}^{cd}{}_{2}^{1}$$

Boxed, Underlined, and Overlined Elements

The purpose for defining boxed, underlined, and overlined elements as distinct constructs is to simplify the task of identifying elements that are to be enclosed in a box or that have an enhanced overline or underline. To identify these elements, simply indicate the beginning of the element with the appropriate start-tag and end it with the end-tag. The tags for each are the following:

Construct	*Start-tag*	*End-tag*
Boxed element	<box>	< / box>
Underlined element	<unl>	< / unl>
Overlined element	<ovl>	< / ovl>

Any of the elements in the Complex built-up constructs group, repeated and in any order, may be included as the content of any of these constructs. Following is an example of a boxed element:

[35] `<box>a + b< / box>` $\boxed{a+b}$

The default style of rule for boxed elements is a single rule. If you wish to specify one of the other styles shown in Table 2, you may do so with the style attribute. An example is shown below.

The tags for underlined and overlined elements should only be used if the over or underline is to be bold, dashed, dotted, a double rule, or a triple rule. For a single rule, the element should be marked up as an accented element, described below. The default style of rule for overlined and underlined elements is bold. For the values of dashed, dotted, double, or triple, refer to Table 2.

An example of each of these elements follows:

[36] `<unl da>a + b< / unl>` $\underline{a+b}$

[37] `<ovl>a + b< / ovl>` $\overline{a+b}$

[38] `<box b>a + b< / box>` $\boxed{a+b}$

Accented Elements

Mathematical formulas often require that an accent mark, diacritical mark, or other symbol be placed above and/or below another character or characters. Such constructs are called accented elements in this manual.

Appendix C of the "Reference Manual" includes a list of entity references for Latin-based (roman) characters with accent and diacritical marks. Those entity references should be used whenever accented Latin characters are required instead of creating your own. However, in those cases when you need an accented character that is not included in that appendix or an accent on a group of characters, you may create it with the accented element construct.

An accented element construct is made up of two components. The first one is the base character or characters being accented, and the second one is the accent itself. No start-tag is needed for the first component, but the second one must have a start-tag. The accented element construct must have an end-tag. One or more elements from the Character string text group and the Special characters/strings group (excluding any that contain superiors or inferiors) are permitted as the content of an accented element component. The tags for accented element and accented element component are **<a>** and **<ac>**, respectively.

The most commonly used accents are shown in Table 3. The entity reference for each accent is composed of the top coordinate followed immediately by the side coordinate. For instance, the entity reference for a macron or overline is **&d1;**.

TABLE 3. Common Mathematical Accents

Name	&d	
Bar (Macron)	¯	1;
Dot	˙	2;
Double dot (Umlaut)	¨	3;
Circumflex (Caret)	^	4;
Tilde	~	5;
Cancellation slash	/	6;
Triple dot	...	7;
Four dots	....	8;
Hacek (Caron)	ˇ	9;
Circle	°	10;
Breve	˘	11;

The placement of the second component with respect to the first in an accented element is determined by an attribute called valign (for "vertical alignment"). Its permitted values are "u" for up (the default), "m" for middle, and "d" for down. Some examples follow:

[39] $\vec{x}$ `<a>x<>&ar;</a>`

[40] $\overrightarrow{xy}$ `<a>xy<>&ar;</a>`

[41] $\vec{\vec{x}}$ `<a<a>x<>&d5;</a><>&ar;</a>`

[42] $\overrightarrow{\widetilde{abc}}$ `<a d><a>abc<>&d5;</a><>&ar;</a>`

[43] or `<a<a d>abc<>&ar;</a><>&d5;</a>`

Three short references have been defined for your use within an accented element construct. They are the following:

In place of	*You may use*
˜	~
ˆ	^
<ac>	|

These characters may be used if they are on your keyboard. Otherwise, the entity reference or tag should be used. Outside the accented element construct, these characters will be considered as text rather than short references.

Now the third example above may be keyed as follows:

[44] $\vec{\tilde{x}}$ `<a/<a/x|~/|&ar;/`

1. Roman Functions

A roman function is a single mathematical symbol consisting of two or more roman letters, typically naming a function. The tag is <rf>; an end-tag may be needed to indicate where the function ends. An empty or null end-tag may be used. Text is the only permitted content of a roman function. Some examples include Tr, csch, sin, cos, tan, log, Re, Im, Hom, and rank. The markup for Tr is the following:

[45] `<rf/Tr/`

2. Increment

This construct is used mainly for differentials. An increment or differential is a single symbol, typically consisting of two letters. The first letter is usually d, $đ$, ∂, δ, Δ, or D. The start-tag is <inc> and the end-tag is required. The content of an increment may be character data, superiors and inferiors (except those that contain elements from the Built-up elements group, the Horizontally-oriented elements group, and the Complex built-up constructs group), and accented elements. Examples follow:

[46] dx `<inc/dx/`

[47] $\delta u'$ `<inc/<g/d/u&j7;/`

3. Generically-coded Characters

Generically-coded symbols are one- or multi-character strings that are usually given a type style other than the customary defaults. Therefore, it is an aid to your publisher if you identify them so they can be treated properly.

The content of a generically-coded character may be character data, superiors and inferiors (except those that contain elements from the Built-up elements group, the Horizontally-oriented elements group, and the Complex built-up constructs group), and accented elements. The end-tag is required for a generically-coded character. The most common classes are the following:

Vector—typically a single character denoting a vector of certain kinds. Not all vectors get special treatment, so the decision must be left to you. Examples follow:

[48] $v_x = (1,0,0)$ `<v/v<inf/x//=(1, 0, 0)`

[49] $\vec{B} \times \vec{V}$ `<v/B/&jx;<v/V/`

Dyadic—this is for fully dyadic, matrix, or rank-2 tensor. It is similar to a vector but with greater emphasis or enhancement. For example,

[50] **A** is keyed as `<dy/A/`

[51] $\vec{\vec{A}}$ is keyed as `<dy/A/`

Field—typically a single capital Latin character, denoting some encompassing set. Examples follow:

[52] $\mathbb{R}^2$ `<fi/R/<sup/2/`

[53] $\mathbb{N}$ `<fi/N/`

This list of special character string elements might not be complete for your requirements. If you need a different one, contact your publisher to agree on a tag for the new element. It will be the responsibility of your publisher to include the declaration statement for the new element in the document type declaration.

Type Style Change Tags

Most of the time, you will not have to be concerned with the type style of the elements and constructs within your mathematical formula, and will not have to mark them up in any special way to get the default type styles. Several elements that diverge from the default are described above in the subsection called Character String Elements. You are encouraged to mark up those elements whenever they occur.

For those situations when you want to explicitly indicate a type style change for an element not included in the aforementioned section, you may use the following tags:

Tag	*Description*
<b>*	bold
<cyr>*	Cyrillic (See Note 1)
<sc>*	script ("fancy")
<ge>*	German (See Note 2)
<g>*	Greek (See Note 3)
<it>*	italic
<op>*	openface ("blackboard boldface")
<rm>*	roman
<ssf>*	sans serif
<ty>*	monospace ("typewriter")
<bsc>*	bold script
<bge>*	bold German (See Note 2)
<bg>*	bold Greek (See Note 3)
<bi>*	bold italic
<bisf>*	bold italic sans serif
<bsf>*	bold sans serif
<isf>*	italic sans serif
<scp>*	small caps

*End-tag required

Note 1: The Cyrillic alphabet and the characters you must type within the Cyrillic start- and end-tags are included in Table 4.

Note 2: The German alphabet (Fraktur)—sometimes called (erroneously) Gothic—is also used for mathematical symbols. The characters are shown in Table 5.

Note 3: The Greek characters used in mathematical formulas and the characters you must type within the mathematical Greek start- and end-tags are included in Table 6.

TABLE 4. Cyrillic Characters

Lowercase		*Uppercase*	
Character	*Keyed*	*Character*	*Keyed*
а	a	А	A
б	b	Б	B
в	v	В	V
г	g	Г	G
д	d	Д	D
е	5	Е	_
ж	1	Ж	!
з	z	З	Z
и	i	И	I
й	j	Й	J
к	k	К	K
л	l	Л	L
м	m	М	M
н	n	Н	N
о	o	О	O
п	p	П	P
р	r	Р	R
с	s	С	S
т	t	Т	T
у	u	У	U
ф	f	Ф	F
х	h	Х	H
ц	q	Ц	Q
ч	c	Ч	C
ш	x	Ш	X
щ	w	Щ	W
ъ	3		
ы	y	Ы	Y
ь	4	Ь	+
э	e	Э	E
ю	6	Ю	?
я	2	Я	%

TABLE 5. German (Fraktur) Characters

Lowercase		*Uppercase*	
Character	*Keyed*	*Character*	*Keyed*
𝔞	a	𝔄	A
𝔟	b	𝔅	B
𝔠	c	ℭ	C
𝔡	d	𝔇	D
𝔢	e	𝔈	E
𝔣	f	𝔉	F
𝔤	g	𝔊	G
𝔥	h	ℌ	H
𝔦	i	ℑ	I
𝔧	j	𝔍	J
𝔨	k	𝔎	K
𝔩	l	𝔏	L
𝔪	m	𝔐	M
𝔫	n	𝔑	N
𝔬	o	𝔒	O
𝔭	p	𝔓	P
𝔮	q	𝔔	Q
𝔯	r	ℜ	R
𝔰	s	𝔖	S
𝔱	t	𝔗	T
𝔲	u	𝔘	U
𝔳	v	𝔙	V
𝔴	w	𝔚	W
𝔵	x	𝔛	X
𝔶	y	𝔜	Y
𝔷	z	ℨ	Z

TABLE 6. Greek Characters

	Lowercase		*Uppercase*	
Name	*Character*	*Keyed*	*Character*	*Keyed*
Alpha	α	a	A	A
Beta	β	b	B	B
Gamma	γ	g	Γ	G
Delta	δ	d	Δ	D
Partial	∂	6		
Del (nabla)			∇	1
Epsilon	ϵ	e	E	E
(Cursive)	ε	3		
Digamma	$\digamma$	5		
Zeta	ζ	z	Z	Z
Eta	η	h	H	H
Theta	θ	q	Θ	Q
(Cursive)	ϑ	j	Θ	J
Iota	ι	i	I	I
Kappa	κ	k	K	K
(Cursive)	$\varkappa$	8		
Lambda	λ	l	Λ	L
Mu	μ	m	M	M
Nu	ν	n	N	N
Xi	ξ	x	Ξ	X
Omicron	o	o	O	O
Pi	π	p	Π	P
	ϖ	2		
Rho	ρ	r	P	R
(Cursive)	ϱ	9		
Sigma	σ	s	Σ	S
(Final)	ς	v		
Tau	τ	t	T	T
Upsilon	υ	u	Υ	U
Phi	ϕ	f	Φ	F
(Cursive)	φ	4		
Chi	χ	c	X	C
Psi	ψ	y	Ψ	Y
Omega	ω	w	Ω	W

The format for using any of the type style tags is that the character(s) or mathematical element requiring a different type style be enclosed between the start and end-tags for the desired type style. Some examples follow:

1. Small caps roman numerals for ionization states in spectroscopy (Fe IV): `Fe <scp / IV /`
2. Roman e for the base of the natural logarithms—Napierian base (to distinguish it from *e* for the electric charge): `<rm / e /`
3. Greek characters: `<g / a /` , `<g / 6 /` will produce α, ∂

Spacing and Positioning Tags

1. Turn Lines

You may want to indicate the particular locations in your formula where a line break should occur. For instance, if an upper or lower limit contains two limits, you would probably want to indicate that the second one should be on a separate line; or, if you know that a construct is going to be long and that an arbitrary line break will interfere with the intended meaning, you might want to indicate where the line break may reasonably occur. In both cases, you may use the turn line tag (`<tu>`) to indicate where the break should be. An example follows:

[54] `<sum>i = 0<tu>n = 1< / ll>i = 5<tu>`

`n = 7< / ul>2n<inf / i / < / sum>` $\sum_{\substack{i=0 \\ n=1}}^{\substack{i=5 \\ n=7}} 2n_i$

A turn line does not have any content of its own and should not have an end-tag.

A turn line may occur almost anywhere in a formula; but it should not be included in the following:

label
operator (in limits)
index (in radicals)
layer (in stacks)
superiors
inferiors
accented elements
functions
increment
vector
dyadic
field

Note that a formula line will automatically begin on a new line; so, it is not necessary to indicate a turn line before a formula line.

2. Mark and Mark Reference

The mark and mark reference tags provide you with the means to align constructs, elements, or characters vertically or horizontally. The desired alignment point is identified with the appropriate mark tag (`<hmk>` for a horizontal mark or `<vmk>` for a vertical mark). In addition to marking the point, you must also assign it a unique identifier so that each mark can be distinguished from all the other

marks you might have. The unique identifier is assigned by an attribute called id. Its value for a horizontal mark is the letters "hm" plus any number that will keep the value unique; and for a vertical mark the value is "vm" plus any number that will keep it unique. For example, the first marked vertical point might be identified with the tag `<vmk id = vm1>` and a second point identified with `<vmk id = vm2>`.

Use of the vertical mark and mark reference is very similar to using a tab. The information following a vertical mark reference tag will have the same left alignment as the information immediately following the vertical mark tag being referenced. Any construct, element, or character that is to be aligned on the same vertical plane with a vertical mark must be immediately preceded with a vertical mark reference tag (`<vmkr>`). The way you identify the particular vertical mark being referenced (even if you have only one mark) is to include an attribute called rid (for reference id) in the vertical mark reference tag. The value of rid must be the id value of the vertical mark being referenced. A vertical mark may be referenced multiple times. The corresponding vertical mark reference tags for those in the examples above are `<vmkr rid = vm1>` and `<vmkr rid = vm2>`.

Use of the horizontal mark reference tag will result in the following information having the same baseline as the information immediately following the horizontal mark tag being referenced. Constructs, elements, or characters that are to be aligned on the same horizontal plane with a horizontal mark follow the same rules as those given for vertical mark. However, the tag for a horizontal mark reference is `<hmkr>`.

The mark tags and the mark reference tags have no content and, therefore, do not require end-tags. They may appear anywhere in a formula.

Before using the mark and mark reference tags, you should check to make sure your publisher supports this feature.

3. Horizontal and Vertical Spacing

The spacing in formulas will, in most cases, be taken care of automatically, such as around mathematical signs, around roman functions and increments, or between the lines of a mathematical formula. In the situations when the spacing must be specified explicitly, it will normally be done by your publisher. However, if you have a requirement for wide or thin spaces and want to indicate their location yourself, you may do so with the horizontal space tag (`<hsp>`) and the vertical space tag (`<vsp>`).

For horizontal spacing the amount of space you require may be specified with an attribute called sp. An sp is approximately the width of an uppercase M. The default value for sp is 2. The value of sp may be given as fractions or multiples of "ems." An example follows:

[55] $y = \sum x_i, \quad \text{where} \quad x_i = r_1^2 \quad \text{for } 1 \leqslant i \leqslant n$

```
<fd>y = <sum><opd>x<inf / i / < / sum>,<hsp><phr / where /
<hsp sp = 1>x<inf / i / = r<sup / 2 / <inf / 1 / <hsp sp = 1>
<phr / for / <hsp sp = .5>1&jf;i&jf;n< / fd>
```

Note that the full tag for operand (`<opd>`) must be included to indicate that the following information is an operand and not an upper or lower limit. The tag `<phr>` is explained under Phrases below.

The amount of vertical space required may be specified by using the sp attribute in the vertical space tag (<vsp>). For this tag, the default value for sp is approximately one-fourth the height of an upper case M.

4. Atom Change Tag

In this manual, an atom refers to a character or a string of characters (hereafter called an expression) that serves a particular function. For example, an equals sign (=) is a relation atom, and a summation sign (Σ) is a large operator atom. Here is a complete list of the different kinds of atoms supported by this application:

ord	is an ordinary atom like "x";
op	is a large operator like "Σ";
bin	is a binary operation atom like "+";
rel	is a relation atom like "=";
open	is an opening atom like "⟨";
close	is a closing atom like "⟩";
punct	is a punctuation atom like ",".

For the atoms that normally serve the functions listed above, the spacing around them is taken care of automatically. In the event that you want to designate a function to an atom that is different from its normal function, you may do so with the atom change tag <ach>.

In addition, you must include an attribute called atom in the tag to indicate the new function of the expression. For example, to change the character x from an ordinary atom to a closing atom, mark it up as follows:

[56] `<ach close / x /`

The atom change element requires an end-tag.

Phrases

Phrases are usually strings of text within mathematics, either as connectives within a displayed formula, e.g.,

$$f(x, y) = 0 \quad \text{if and only if} \quad x = 0 \text{ or } y = 0$$

or as verbose mathematical symbols, e.g.,

$$S \propto \sqrt{\text{word length}}, \qquad \text{force} = \text{mass} \times \text{acceleration}$$

A phrase may be composed of any of the elements in the Nonmath text group—in any order and repeatable if desired. For example, the content of an array cell might be a list or a paragraph. The tag for a phrase is <phr>.

The two examples above might be marked up as follows:

```
[57] <fd>f(x,y) = 0<hsp sp = 1><phr / if and only if /
     <hsp sp = 1>x = 0<hsp sp = .5><phr / or / <hsp sp = .5>
     y = 0< / fd>

[58] or: <fd>f(x,y) = 0<phr / if and only if / x = 0<phr / or /
         y = 0< / fd>
```



```
[59] <fd>S &vi;<rad><phr / word length,/ < / rad><hsp>
     <phr / force / = <phr / mass / &jx;<phr / acceleration / < / fd>

[60] or:
<fd>S &vi;<rad><phr / word length,/ < / rad><phr / force / =
<phr / mass / &jx;<phr / acceleration / < / fd>
```

Reference to a formula label is done frequently in the text of a manuscript or within phrases inside a formula. Most of the time such a reference does not require special identification. However, if the manuscript will be included in a database in the future or if a publisher gives special treatment to references to formula labels, it is necessary to identify the references with a tag.

The tag <lar> is the one to use when identifying a reference to a label that has been identified with the tag <la>. The end-tag for a label reference is required. As an example, assume that the label 1.1 has been properly identified with the label tag. Then, it would be referenced in the text of your manuscript or within a phrase in a formula as follows:

[61] `<lar / 1.1 /` or `<lar>1.1< / >`

The other reference tags in the Nonmath text group may be used in the same way within a mathematical phrase or the text of your manuscript.

APPENDIXES

APPENDIX A

FORMAL SYNTAX FOR MATHEMATICAL FORMULAS

Contents:

Document Type Declarations
Public Entity Declarations
Data Content Notations
Incorporation of Mathematical Formulas into Other Documents
Mathematical Formulas as Stand-alone Documents

This appendix contains the formal syntax for implementing the AAP's markup for mathematical formulas. It is intended for use by publishers and/or software implementers.

DOCUMENT TYPE DECLARATION FOR DISPLAYED MATHEMATICAL FORMULAS

<!-- Public declarations for AAP mathematical and scientific displayed formula notation. Typical invocation within the DTD of your selected document type:

```
    <!ENTITY % aapmth1 PUBLIC "-//USA/AAP//DTD MATH-1//EN">
    %aapmth1;
-->
<!ENTITY % commth2 PUBLIC "-//USA/AAP//DTD COMMATH-2//EN"
            -- Common declarations set 2 for math formulas -->
<!ENTITY % commth1 PUBLIC "-//USA/AAP//ENTITIES COMMATH-1//EN"
            -- Common declarations set 1 for math formulas -->
%commth1;
<!ENTITY % form1 "fd" -- Element type is Geometric Formula -->
<!NOTATION geo.form PUBLIC "-//USA/AAP//NOTATION Geometric For-
mula Notation//EN"
            -- SPACE, TAB, RS, and RE are ignored as data.
               Characters are classified as ord, op, bin,
```

```
                rel, open, close, or punct atoms, unless
                reclassified by the atom change tags. -->
<!--ELEMENT        MIN CONTENT   EXCEPTIONS
             ATTRIBUTE   VALUE                    DEFAULT -->
<!ELEMENT %form1;   - -   (la?, la?, fl)+>
<!ATTLIST %form1; dcn     NOTATION (geo.form)     geo.form>
<!ELEMENT fl            0 0  (%f-butxt;)*>
<!ELEMENT la            - 0  (%f-cstxt;|%f-ph;)*
                             -(tu)>
<!ATTLIST la  loc         (pre|post)            #IMPLIED
              id          ID                    #IMPLIED
              -- Label, reference to (<lar>) should be defined in
                 the incorporating document type definition. -->
<!USEMAP      formmap %form1;>
%commth2;
```

DOCUMENT TYPE DECLARATION FOR IN-LINE MATHEMATICAL FORMULAS

<!-- Public declarations for AAP mathematical and scientific in-line formula notation. Typical invocation within the DTD of your selected document type:

```
   <!ENTITY % aapmth2 PUBLIC "-//USA/AAP//DTD MATH-2//EN">
   %aapmth2;
-->
<!ENTITY % commth2 PUBLIC "-//USA/AAP//ENTITIES COMMATH-2//EN"
              -- Common declarations set 2 for math formulas -->
<!ENTITY % commth1 PUBLIC "-//USA/AAP//ENTITIES COMMATH-1//EN"
              -- Common declarations set 1 for math formulas -->
%commth1;
<!ENTITY % form2 "f" -- Element type is Geometric Formula -->
<!NOTATION geo.form PUBLIC "-//USA/AAP//NOTATION Geometric For-
mula Notation//EN"
          -- SPACE, TAB, RS, and RE are ignored as data.
             Characters are classified as ord, op, bin,
             rel, open, close, or punct atoms, unless
             reclassified by the atom change tags. -->
<!--ELEMENT     MIN  CONTENT    EXCEPTIONS
           ATTRIBUTE  VALUE                DEFAULT -->
<!ELEMENT %form2;     - - (%f-butxt;)*>
<!ATTLIST %form2;  dcn   NOTATION (geo.form)      geo.form>
<!USEMAP formmap %form2;>
%commth2;
```

COMMON DECLARATIONS FOR AAP MATHEMATICAL FORMULAS

Common Declarations, Set 1

<!-- Public entity for common declarations, set 1, for AAP mathematical formulas. Typical invocation:

```
   <!ENTITY % commth1 PUBLIC "-//USA/AAP//ENTITIES COMMATH-1//
EN">
   %commth1;
-->
<!ENTITY % p.fnt.ph   "sc|op|ssf|ty|bsc|bisf|bsf|isf"
                                  -- Type style elements -->
<!ENTITY % sp.pos     "hmk|hmkr|vsp|hsp|tu|zw|vmk|vmkr"
                        -- spacing and positioning tags -->
<!ENTITY % f-cs       "a|(%p.em.ph;)|(%p.fnt.ph;)|g|bg|(%sp.pos;)"
                                  -- Character strings -->
<!ENTITY % f-cstxt    "#PCDATA|%f-cs;" -- Character string text
-->
<!ENTITY % f-scs      "rf|inc|v|dy|fi"
                          -- Special characters or strings -->
<!ENTITY % limits     "pr|in|sum" -- Special limits elements -->
<!ENTITY % f-bu       "fr|rad|lim|ar|fen" -- Built-up elements -->
<!ENTITY % f-ph       "box|unl|ovl|sup|inf" -- Horizontal elements
-->
<!ENTITY % f-butxt    "(%f-bu;)|(%limits;)|(%f-cstxt;)|(%f-scs;)
                       |(%f-ph;) -- Built-up element text -->
```

Common Declarations, Set 2

<!-- Public entity for common declarations, set 2, for AAP mathematical formulas. Typical invocation:

```
   <!ENTITY % commth2 PUBLIC "-//USA/AAP//DTD COMMATH-2//EN">
   %commth2;
-->
<!-- Entities for short references -->
<!ENTITY  fen-lsq   STARTTAG "fen sqb">
<!ENTITY  fen-lbr   STARTTAG "fen cub">
<!ENTITY  fen-lpa   STARTTAG "fen par">
<!ENTITY  fen-rsq           "<rp sqb></fen>">
<!ENTITY  fen-rbr           "<rp cub></fen>">
<!ENTITY  fen-rpa           "<rp par></fen>">
<!ENTITY  arr-row           "<arr<arc>">
<!ENTITY  arr-cell  STARTTAG "arc">
<!ENTITY  tildref           "&tilde;">
<!ENTITY  circref           "&circ;">
<!ENTITY  ac-comp   STARTTAG "ac">
<!SHORTREF formmap
                    -- Map for general use in formulas --
      "[" fen-lsq  -- Left square bracket: <fen sqb> --
      "{" fen-lbr  -- Left brace: <fen cub> --
      "(" fen-lpa  -- Left parenthesis: <fen par> --
>
<!SHORTREF fencemap
                    -- Map for fence constructs --
      "[" fen-lsq  -- Left square bracket: <fen sqb> --
      "{" fen-lbr  -- Left brace: <fen cub> --
      "(" fen-ipa  -- Left parenthesis: <fen par> --
```

```
      "]"  fen-rsq -- Rt square bracket: <rp
                      sqb></fen> --
      "}"  fen-rbr -- Rt brace: <rp cub></fen> --
      ")"  fen-rpa -- Rt paren: <rp par></fen> --
>
<!SHORTREF arrmap
                   -- Map for array constructs --
      "@"  arr-row -- Commercial at symbol: <arr<arc> --
      "|"  arr-cell-- Vertical bar: <arc> --
>
<!SHORTREF accmap
                   -- Map for accented elements --
      "~"  tildref -- tilde: &tilde; --
      "^"  circref -- circumflex: &circ; --
      "|"  ac-comp -- accented element component --
>
<!ENTITY % f-phtxt "p|(%p.em.ph;)|(%p.lst.d;)|(%p.rf.ph;)|q|#PCDATA"
                                -- elements for nonmath text -->
<!ENTITY % f-align "align (l|c|r) c"
                                 -- Align attribute (center) -->
<!ENTITY % f-post  "par|sqb|llsqb|rrsqb|cub|ceil|fl|ang
                                |sol|vb|uc|dc" -- Brackets -->
<!ENTITY % f-style "s|d|t|da|dot|b|bl|n" -- Styles -->
<!ENTITY % f-atom  "ord|op|bin|rel|open|close|punct"
                   -- atoms for changing the function of
                      a character or an expression -->
<!ENTITY % mthmap1 PUBLIC "-//USA/AAP//ENTITIES MATHMAP-1//EN"
        -- mapping entity declarations for all math symbols -->
%mthmap1;
<!NOTATION greek2   PUBLIC "-//USA/AAP//NOTATION GREEK-2//EN"
                                -- Greek characters for math -->
<!NOTATION german1  PUBLIC "-//USA/AAP//NOTATION GERMAN-1//EN"
                               -- German characters for math -->
<!--ELEMENT              MIN   CONTENT    EXCEPTIONS
          ATTRIBUTE   VALUE                      DEFAULT -->

<!--                          Built-up elements -->
<!ELEMENT fr              - -  (nu, de)>
<!ATTLIST fr shape        (built|case|sol)          #IMPLIED
          %f-align;
          -- Alignment of narrower of nu or de
             with respect to fraction bar. --
          style       (%f-style;)                 s
          -- Style of fraction bar. -->
<!ELEMENT (nu|de)         0 0  (%f-butxt;)*>
<!ELEMENT lim             - -  (op, ll, ul, opd?)>
<!ATTLIST lim align       (r|c)                     #IMPLIED>
<!ELEMENT op              0 0  (%f-cstxt;|rf|%f-ph;)
                                        -(tu)>
<!ELEMENT (ll|ul)         0 0  (%f-butxt;)*>
<!ELEMENT opd             - 0  (%f-butxt;)*>
<!ELEMENT (%limits;)      - -  (ll, ul, opd?)>
<!ATTLIST (%limits;) align  (r|c)                   #IMPLIED>
<!ELEMENT rad             - -  (rcd, rdx?)>
<!ELEMENT rcd             0 0  (%f-butxt;)*>
<!ELEMENT rdx             - 0  (%f-butxt;)* -(tu)>
```



```
<!ELEMENT fen             - -  ((%f-butxt;)*,
                                  (cp, (%f-butxt;)*)*, rp)>
<!ATTLIST fen lp          (%f-post;)                vb
          style           (%f-style;)               s>
<!USEMAP fencemap         fen>
<!ELEMENT (cp|rp)         - 0  EMPTY>
<!ATTLIST (cp|rp)) post   (%f-post;)                vb
          style           (%f-style;)               s>
<!ELEMENT ar              - -  (arr+)>
<!ATTLIST ar    cs              NMTOKENS                "A bl"
          -- Valid NMTOKENS are those in %f-style; they indicate
             separator style. Specify from 1 to F+1 style tokens,
             where F is the number of columns, each preceded by
             a column number from 0 through F. 0 applies to
             separator before 1st column, and 1 through F to
             separators after corresponding columns; unspeci-
             fied separators are blank. Special column "numbers"
             F (for final) and A (for all others) can be used.
             Examples:
                 "0 d F d" gives double rules as borders.
                 "1 b"     gives bold rule after first column.
          --
             rs           NMTOKENS                   "A bl"
          -- Value specified as for cs, but applies to row
             separators. --
             ca           NMTOKENS                   "A c"
          -- Specify from 1 to F alignment tokens
              ("l"|"c"|"r"|"e"|"d"),
             where F is the number of columns, to indicate align-
             ment of cells within the corresponding column. Align-
             ments are: Left, Center, Right, Exponent, and Decimal
             point. Default is "c" alignment for any unspecified
             column. Tokens are paired with column numbers, as
             with cs. Examples:
             "1 l A r"   gives left aligned 1st column,
                         all others right.
             "1 l"       gives left aligned 1st column,
                         all others center.
          -->
<!USEMAP arrmap ar>
<!ELEMENT arr             - 0  (arc+)>
<!ELEMENT arc             0 0  (%f-butxt;)*>
<!ATTLIST arc align       (c|r|l|e|d)               #IMPLIED
          -- Alignment specified for a cell overrides that
             specified for the column and is in effect for that
             cell only. -->
<!ELEMENT stk             - -  (lyr+)>
<!ELEMENT lyr             0 0  (%f-butxt;)* -(tu)>
<!ATTLIST lyr %f-align;
          -- Alignment of layer with respect to previous
             layer in stack. -->
<!ELEMENT ach             - -  (%f-butxt;)*
          -- For associating an atom with any expression. -->
<!ATTLIST ach atom        (%f-atom;)                #REQUIRED
          -- Atom is implied by content notation if not
             specified. -->
```

```
<!--                        Phrases -->
<!ELEMENT (sup|inf)         - - (%f-butxt;)* -(tu)>
<!ATTLIST (sup|inf) loc     (pre|post)                  post
          -- Alignment of superior/inferior with respect to
             the preceding character or contextual element.
             They cannot follow a start-tag. Successive
             superior/inferior elements apply to same subject
             character or element, not to one another.
             For A to the X-cubed: A<sup/X<sup/3//
             For staggered ``post'' scripts, insert a zero
             width element: T<inf/1/<zw><sup/2/<zw><inf/3/
          -->
<!ELEMENT zw                - O  EMPTY>
<!ELEMENT box               - -  (%f-butxt;)*>
<!ATTLIST box style         (%f-style;)                  s>
<!ELEMENT (unl|ovl)         - -  (%f-butxt;)*>
<!ATTLIST (unl|orl) style   (%f-style;)                  b>

<!--                        Character strings -->
<!ELEMENT a                 - -  (ac, ac) -(tu)>
<!ATTLIST a valign          (u|m|d)                      u
          -- Vertical alignment of embellishment with respect
             to subject. -->
<!USEMAP accmap a>
<!ELEMENT ac                O O  (%f-cstxt;|%f-scs;)*
                                      -(sup|inf)>
<!ELEMENT (%f-scs;)         - O  ((%f-cstxt;)|sup|inf)*
                                 -(tu|%limits;|%f-bu;|%f-ph;)>
<!ELEMENT phr               - O  (%f-phtxt;)*
          -- Nonmath (ordinary) text -->

<!--                      Location elements -->
<!ELEMENT (hmk|vmk)       - O  EMPTY
          -- Notes horizontal or vertical location -->
<!ATTLIST (hmk|vmk) id    ID                           #REQUIRED>
<!ELEMENT (hmkr|vmkr)     - O  EMPTY
          -- Aligns current location with previously marked
             horizontal or vertical location. -->
<!ATTLIST (hmkr|vmkr) rid  IDREF                       #REQUIRED>
<!ELEMENT hsp             - O  EMPTY>
<!ATTLIST hsp sp          NUTOKEN                      2
          -- Adjusts location by specified (positive or negative)
             offset in em spaces and fractions of an em
             (e.g., -.3, 1.2).
          -->
<!ELEMENT vsp             - O  EMPTY>
<!ATTLIST vsp sp          NUTOKEN                      .25
          -- Adjusts location by specified (positive or negative)
             offset in em spaces and fractions of an em
             (e.g., -.3, 1.2).
          -->
<!ELEMENT tu              - O  EMPTY>
<!ELEMENT (g|bg)          - -  (#PCDATA)>
<!ATTLIST (g|bg) lang     NOTATION (greek2)            greek2>
<!ELEMENT (%p.fnt.ph;)    - -  (%f-cstxt;)*>
<!ELEMENT (ge|bge)        -- (#PCDATA)>
<!ATTLIST (ge|bge) lang   NOTATION (german1)           german1>
```


PUBLIC ENTITY DECLARATIONS

This section contains the public entity declarations, as they should appear in an AAP document type declaration, for the set of public entity references defined for use with the AAP's Electronic Manuscript Series: *Markup of Mathematical Formulas.* In some cases, the public entity sets are defined in the SGML Standard and are included herein by reference only. Also, some of the declarations are contained in Appendix B—Part 2 of the AAP Standard and are so indentified in this section. Any entity declaration should appear only once in a given document type declaration.

1. For the roman characters with accents and diacritical marks defined in ISO International Standard 6937/2 (Information processing—Coded character sets for text communication—Part 2: Latin alphabetic and non-alphabetic graphic characters):

   Refer to AAP Standard, Part 2, Appendix B, Item 1.

2. For the set of accent and diacritical marks as separate characters defined in ISO International Standard 6937/2:

   Refer to AAP Standard, Part 2, Appendix B, Item 2.

3. For the general use numeric and special graphic characters defined in ISO International Standard 6937/2:

   Refer to AAP Standard, Part 2, Appendix B, Item 5.

4. For the general use publishing characters, including those proposed in ISO Draft Proposed Standard 6937/6:

   Refer to AAP Standard, Part 2, Appendix B, Item 6.

5. For the general technical symbols, including those proposed in ISO Draft Proposed Standard 6937/5:

   Refer to AAP Standard, Part 2, Appendix B, Item 7.

6. For additional mathematical symbols, miscellaneous (ordinary):

```
<!ENTITY %f ams-m PUBLIC "ISO 8879-1986//ENTITIES
Added Math Symbols: Ordinary//EN"
                        -- miscellaneous mathematical symbols -->
%ams-m;
```

7. For additional mathematical symbols that are binary and large operators:

```
<!ENTITY % ams-bl PUBLIC "ISO 8879-1986//ENTITIES
Added Math Symbols: Binary Operators//EN"
                          -- binary and large operators -->
%ams-bl;
```

8. For additional mathematical symbols that indicate relations:

```
<!ENTITY % ams-r PUBLIC "ISO 8879-1986//ENTITIES
Added Math Symbols: Relations//EN" -- relations -->

%ams-r;
```

9. For additional mathematical symbols that indicate negated relations:

```
<!ENTITY % ams-n PUBLIC "ISO 8879-1986//ENTITIES
Added Math Symbols: Negated Relations//EN"
                                -- negated relations -->

%ams-n;
```

10. For additional mathematical symbols that indicate arrow relations:

```
<!ENTITY % ams-a PUBLIC "ISO 8879-1986//ENTITIES
Added Math Symbols: Arrow Relations//EN"
                                -- arrow relations -->

%ams-a;
```

11. For additional mathematical symbols that function as closing and opening delimiters:

```
<!ENTITY % ams-co PUBLIC "ISO 8879-1986//ENTITIES
Added Math Symbols: Delimiters//EN"
                        -- closing and opening delimiters -->

%ams-co;
```

12. Additional miscellaneous entity declarations:

```
<!ENTITY tprime  SDATA "[triple prime]" -- triple prime
-->
<!ENTITY dast    SDATA "[down ast]" -- downward asterisk
-->
<!ENTITY cansla  SDATA "[cancel slash]" -- cancellation
slash -->
<!ENTITY tripdot SDATA "[triple dot]" -- triple dot -->
<!ENTITY fourdot SDATA "[four dots]" -- four dots -->
```

13. Entity declarations mapping table form of special symbol entity references to their standard form:

```
<!ENTITY % mthmap1 PUBLIC "-//USA/AAP//ENTITIES MATH-
MAP-1//EN" -- mapping declarations -->

%mthmap1;
```

This public entity set includes the following declarations:

```
<!ENTITY d1   "&macr;">        <!ENTITY mr   "&there4;">
<!ENTITY d2   "&dot;">         <!ENTITY mg   "&becaus;">
<!ENTITY d3   "&uml;">         <!ENTITY me   "&isin;">
<!ENTITY d4   "&circ;">        <!ENTITY le   "&ni;">
<!ENTITY d5   "&tilde;">       <!ENTITY vb   "&sube;">
<!ENTITY d6   "&cansla;">      <!ENTITY ve   "&supe;">
<!ENTITY d7   "&tripdot;">     <!ENTITY a23  "&iff;">
<!ENTITY d8   "&fourdot;">     <!ENTITY p18  "&not;">
<!ENTITY d9   "&caron;">       <!ENTITY ja   "&forall;">
<!ENTITY d10  "&ring;">        <!ENTITY jb   "&exist;">
<!ENTITY d11  "&breve;">       <!ENTITY ji   "&int;">
<!ENTITY j26  "&plusmn;">      <!ENTITY li   "&conint;">
<!ENTITY j13  "&divide;">      <!ENTITY je   "&infin;">
<!ENTITY jx   "&times;">       <!ENTITY jt   "&sim;">
<!ENTITY j10  "&dot;">         <!ENTITY m28  "&ap;">
<!ENTITY al   "&larr;">        <!ENTITY v1   "&sime;">
<!ENTITY ar   "&rarr;">        <!ENTITY v18  "&cong;">
<!ENTITY au   "&uarr;">        <!ENTITY vn   "&le;">
<!ENTITY ad   "&darr;">        <!ENTITY jn   "&ne;">
<!ENTITY ma   "&minus;">       <!ENTITY vo   "&ge;">
<!ENTITY j5   "&dagger;">      <!ENTITY vi   "&prop;">
<!ENTITY j4   "&Dagger;">      <!ENTITY la   "&aleph;">
<!ENTITY j12  "&malt;">        <!ENTITY ms   "&square;">
<!ENTITY l6   "&clubs;">       <!ENTITY vq   "&par;">
<!ENTITY l1   "&diams;">       <!ENTITY lp   "&bottom;">
<!ENTITY l2   "&hearts;">      <!ENTITY vp   "&perp;">
<!ENTITY l5   "&spades;">      <!ENTITY m6   "&angsph;">
<!ENTITY m1   "&loz;">         <!ENTITY l4   "&ang90;">
<!ENTITY m10  "&utri;">        <!ENTITY j19  "&lang;">
<!ENTITY m11  "&dtri;">        <!ENTITY j20  "&rang;">
<!ENTITY m13  "&rtri;">        <!ENTITY j7   "&prime;">
<!ENTITY m12  "&ltri;">        <!ENTITY j8   "&Prime;">
<!ENTITY lm   "&flat;">        <!ENTITY jm   "&mnplus;">
<!ENTITY p1   "&lozf;">        <!ENTITY j9   "&tprime;">
<!ENTITY l0   "&bull;">        <!ENTITY j31  "&dast;">
<!ENTITY ls   "&squf;">        <!ENTITY m4   "&ang;">
<!ENTITY l10  "&utrif;">       <!ENTITY m5   "&angmsd;">
<!ENTITY l11  "&dtrif;">       <!ENTITY lb   "&beth;">
<!ENTITY l12  "&ltrif;">       <!ENTITY j6   "&bprime;">
<!ENTITY l13  "&rtrif;">       <!ENTITY mc   "&comp;">
<!ENTITY mk   "&natur;">       <!ENTITY ld   "&daleth;">
<!ENTITY ln   "&sharp;">       <!ENTITY mo   "&empty;">
<!ENTITY j28  "&equiv;">       <!ENTITY lh   "&gimel;">
<!ENTITY mv   "&or;">          <!ENTITY np   "&nexist;">
<!ENTITY my   "&and;">         <!ENTITY jh   "&plank;">
<!ENTITY mw   "&cap;">         <!ENTITY mp   "&weierp;">
<!ENTITY mu   "&cup;">         <!ENTITY l7   "&amalg;">
<!ENTITY vc   "&sub;">         <!ENTITY ks   "&Barwed;">
<!ENTITY vf   "&sup;">         <!ENTITY m15  "&barwed;">
<!ENTITY am   "&lArr;">        <!ENTITY kw   "&Cap;">
<!ENTITY as   "&rArr;">        <!ENTITY ku   "&Cup;">
<!ENTITY jq   "&cuvee;">       <!ENTITY v5   "&gap;">
<!ENTITY ka   "&cuwed;">       <!ENTITY kg   "&gdot;">
<!ENTITY m2   "&diam;">        <!ENTITY vx   "&gE;">
<!ENTITY jy   "&divonx;">      <!ENTITY l9   "&gel;">
<!ENTITY kt   "&intcal;">      <!ENTITY v9   "&gEl;">
<!ENTITY ky   "&lthree;">      <!ENTITY jo   "&ges;">
<!ENTITY km   "&ltimes;">      <!ENTITY vg   "&Gg;">
<!ENTITY l18  "&minusb;">      <!ENTITY m9   "&gl;">
<!ENTITY l31  "&oast;">        <!ENTITY vz   "&gsim;">
```

```
<!ENTITY m16  "&ocirc;">
<!ENTITY m17  "&odash;">
<!ENTITY m2   "&odot;">
<!ENTITY m18  "&ominus;">
<!ENTITY m26  "&oplus;">
<!ENTITY m19  "&osol;">
<!ENTITY mx   "&otimes;">
<!ENTITY l26  "&plusb;">
<!ENTITY l25  "&plusdo;">
<!ENTITY kx   "&rthree;">
<!ENTITY kn   "&rtimes;">
<!ENTITY lz   "&sdotb;">
<!ENTITY l30  "&setmn;">
<!ENTITY l15  "&sqcap;">
<!ENTITY l14  "&sqcup;">
<!ENTITY m21  "&sstarf;">
<!ENTITY lx   "&timesb;">
<!ENTITY mt   "&top;">
<!ENTITY lu   "&uplus;">
<!ENTITY m22  "&wreath;">
<!ENTITY k2   "&xdtri;">
<!ENTITY k3   "&xutri;">
<!ENTITY v2   "&ape;">
<!ENTITY v28  "&asymp;">
<!ENTITY v0   "&bcong;">
<!ENTITY ko   "&bowtie;">
<!ENTITY lt   "&bsim;">
<!ENTITY v23  "&bsime;">
<!ENTITY vu   "&bump;">
<!ENTITY vt   "&bumpe;">
<!ENTITY k10  "&circe;">
<!ENTITY k13  "&colone;">
<!ENTITY v24  "&cuepr;">
<!ENTITY v25  "&cuesc;">
<!ENTITY v16  "&cupre;">
<!ENTITY p26  "&dashv;">
<!ENTITY vh   "&ecir;">
<!ENTITY k12  "&ecolon;">
<!ENTITY vs   "&eDot;">
<!ENTITY k11  "&edot;">
<!ENTITY j17  "&efDot;">
<!ENTITY lo   "&egs;">
<!ENTITY lf   "&els;">
<!ENTITY k17  "&erDot;">
<!ENTITY kj   "&fork;">
<!ENTITY j15  "&frown;">
<!ENTITY nx   "&gnE;">
<!ENTITY nz   "&gnsim;">
<!ENTITY n4   "&lnap;">
<!ENTITY nw   "&lnE;">
<!ENTITY nm   "&lne;">
<!ENTITY lg   "&Gt;">
<!ENTITY v4   "&lap;">
<!ENTITY kl   "&ldot;">
<!ENTITY vw   "&lE;">
<!ENTITY v8   "&lEg;">
<!ENTITY l8   "&leg;">
<!ENTITY jf   "&les;">
<!ENTITY m8   "&lg;">
<!ENTITY vl   "&Ll;">
<!ENTITY vy   "&lsim;">
<!ENTITY ll   "&Lt;">
<!ENTITY k6   "&ltrie;">
<!ENTITY vv   "&mid;">
<!ENTITY v27  "&models;">
<!ENTITY v14  "&pr;">
<!ENTITY v12  "&prap;">
<!ENTITY jj   "&pre;">
<!ENTITY n10  "&prsim;">
<!ENTITY k7   "&rtrie;">
<!ENTITY k5   "&samalg;">
<!ENTITY v15  "&sc;">
<!ENTITY v13  "&scap;">
<!ENTITY v29  "&sccue;">
<!ENTITY jk   "&sce;">
<!ENTITY v11  "&scsim;">
<!ENTITY k15  "&sfrown;">
<!ENTITY jv   "&smid;">
<!ENTITY j14  "&smile;">
<!ENTITY jp   "&spar;">
<!ENTITY v19  "&sqsub;">
<!ENTITY v21  "&sqsube;">
<!ENTITY v20  "&sqsup;">
<!ENTITY v22  "&sqsupe;">
<!ENTITY k14  "&ssmile;">
<!ENTITY kc   "&Sub;">
<!ENTITY va   "&subE;">
<!ENTITY kf   "&Sup;">
<!ENTITY vd   "&supE;">
<!ENTITY vr   "&trie;">
<!ENTITY v3   "&twixt;">
<!ENTITY v26  "&vdash;">
<!ENTITY vj   "&Vdash;">
<!ENTITY m14  "&veebar;">
<!ENTITY vm   "&Vvdash;">
<!ENTITY n5   "&gnap;">
<!ENTITY no   "&gne;">
<!ENTITY ab   "&dharl;">
<!ENTITY ac   "&dharr;">
<!ENTITY a24  "&Larr;">
<!ENTITY az   "&lArr;">
<!ENTITY a11  "&larr2;">
```



```
<!ENTITY ny   "&lnsim;">
<!ENTITY nu   "&nap;">
<!ENTITY n18  "&ncong;">
<!ENTITY n28  "&nequiv;">
<!ENTITY k19  "&ngE;">
<!ENTITY n19  "&nge;">
<!ENTITY n9   "&nges;">
<!ENTITY ng   "&ngt;">
<!ENTITY n20  "&nle;">
<!ENTITY k20  "&nlE;">
<!ENTITY n8   "&nles;">
<!ENTITY n21  "&nlt;">
<!ENTITY n22  "&nltri;">
<!ENTITY k8   "&nltrie;">
<!ENTITY nv   "&nmid;">
<!ENTITY nq   "&npar;">
<!ENTITY nh   "&npr;">
<!ENTITY nj   "&npre;">
<!ENTITY n23  "&nrtri;">
<!ENTITY k9   "&nrtrie;">
<!ENTITY ni   "&nsc;">
<!ENTITY nk   "&nsce;">
<!ENTITY nt   "&nsim;">
<!ENTITY n1   "&nsime;">
<!ENTITY kv   "&nsmid;">
<!ENTITY kp   "&nspar;">
<!ENTITY nc   "&nsub;">
<!ENTITY n16  "&nsube;">
<!ENTITY n24  "&nsubE;">
<!ENTITY nf   "&nsup;">
<!ENTITY n25  "&nsupE;">
<!ENTITY n17  "&nsupe;">
<!ENTITY n26  "&nvdash;">
<!ENTITY n27  "&nvDash;">
<!ENTITY n2   "&nVDash;">
<!ENTITY n0   "&nVdash;">
<!ENTITY n12  "&prnap;">
<!ENTITY n14  "&prnE;">
<!ENTITY n10  "&prnsim;">
<!ENTITY n13  "&scnap;">
<!ENTITY n15  "&scnE;">
<!ENTITY n11  "&scnsim;">
<!ENTITY nb   "&subne;">
<!ENTITY na   "&subnE;">
<!ENTITY ne   "&supne;">
<!ENTITY nd   "&supnE;">
<!ENTITY a14  "&cularr;">
<!ENTITY a15  "&curarr;">
<!ENTITY ae   "&dArr;">
<!ENTITY a9   "&darr2;">
<!ENTITY aj   "&larrhk;">
<!ENTITY a26  "&larrlp;">
<!ENTITY a22  "&larrtl;">
<!ENTITY a16  "&lhard;">
<!ENTITY ak   "&lharu;">
<!ENTITY a3   "&hArr;">
<!ENTITY a2   "&harr;">
<!ENTITY ah   "&lrarr2;">
<!ENTITY an   "&harrw;">
<!ENTITY af   "&lrhar2;">
<!ENTITY a29  "&lsh;">
<!ENTITY at   "&map;">
<!ENTITY jr   "&mumap;">
<!ENTITY a6   "&nearr;">
<!ENTITY nm   "&nlArr;">
<!ENTITY nl   "&nlarr;">
<!ENTITY n3   "&nhArr;">
<!ENTITY j2   "&nharr;">
<!ENTITY nr   "&nrarr;">
<!ENTITY ns   "&nrArr;">
<!ENTITY a4   "&nwarr;">
<!ENTITY a12  "&olarr;">
<!ENTITY a13  "&orarr;">
<!ENTITY a25  "&Rarr;">
<!ENTITY ay   "&rAarr;">
<!ENTITY a10  "&rarr2;">
<!ENTITY ai   "&rlarr2;">
<!ENTITY ag   "&rlhar2;">
<!ENTITY ap   "&rarrhk;">
<!ENTITY a27  "&rarrlp;">
<!ENTITY a21  "&rarrtl;">
<!ENTITY ao   "&rarrw;">
<!ENTITY a17  "&rhard;">
<!ENTITY aq   "&rharu;">
<!ENTITY a30  "&rsh;">
<!ENTITY a7   "&dlarr;">
<!ENTITY a5   "&drarr;">
<!ENTITY av   "&uArr;">
<!ENTITY a8   "&uarr2;">
<!ENTITY a1   "&vArr;">
<!ENTITY a0   "&varr;">
<!ENTITY aw   "&uharl;">
<!ENTITY ax   "&uharr;">
<!ENTITY a23  "&xhArr;">
<!ENTITY jd   "&rpargt;">
<!ENTITY jc   "&lpargt;">
```

DATA CONTENT NOTATIONS

This section contains the data content notations defined for the AAP mathematical formulas application. It includes the notation for mathematical Greek characters and a reference for the notation for Cyrillic characters.

DATA CONTENT NOTATION FOR GREEK CHARACTERS WITHIN MATHEMATICAL FORMULAS

The following is the declaration for the Greek data content notation within mathematical formulas:

```
<!NOTATION greek2 PUBLIC "-//USA/AAP//NOTATION
GREEK-2//EN">
```

For mathematical Greek characters, identify the start of the character(s) with the start-tag for mathematical Greek (i.e., <g>) and end it with the end-tag. For the content of the mathematical Greek element, the characters in the "Keyed" column must be used to represent the corresponding Greek characters as shown in Table A, on p. 57.

DATA CONTENT NOTATION FOR CYRILLIC CHARACTERS WITHIN MATHEMATICAL FORMULAS

The data content notation for Cyrillic characters used within mathematical formulas is the same as that defined in the AAP's *Standard for Electronic Manuscript Preparation and Markup.* Refer to Part 2—Appendix C in that document for the syntax of the Cyrillic notation. The tag for the Cyrillic notation is <cyr>; the end-tag is required.

DATA CONTENT NOTATION FOR GERMAN CHARACTERS WITHIN MATHEMATICAL FORMULAS

The following is the declaration for the German data content notation within mathematical formulas:

```
<!NOTATION german1 PUBLIC "-//USA/AAP//NOTATION
German-1//EN">
```

For mathematical German characters, identify the start of the character(s) with the start-tag for mathematical German (i.e., <ge>) and end it with the end-tag. For the content of the mathematical German element, the characters in the "Keyed" column must be used to represent the corresponding German characters as shown in Table B on p. 58.

INCORPORATION OF MATHEMATICAL FORMULAS INTO OTHER DOCUMENTS

Mathematical formulas (displayed and in-line) may be incorporated into other AAP document types using the following set of declarations in the document type

TABLE A. Greek Characters

	Lowercase		*Uppercase*	
Name	*Character*	*Keyed*	*Character*	*Keyed*
Alpha	α	a	A	A
Beta	β	b	B	B
Gamma	γ	g	Γ	G
Delta	δ	d	Δ	D
Partial	∂	6		
Del (nabla)			∇	1
Epsilon	ϵ	e	E	E
(Cursive)	ε	3		
Digamma	$\digamma$	5		
Zeta	ζ	z	Z	Z
Eta	η	h	H	H
Theta	θ	q	Θ	Q
(Cursive)	ϑ	j	Θ	J
Iota	ι	i	I	I
Kappa	κ	k	K	K
(Cursive)	$\varkappa$	8		
Lambda	λ	l	Λ	L
Mu	μ	m	M	M
Nu	ν	n	N	N
Xi	ξ	x	Ξ	X
Omicron	o	o	O	O
Pi	π	p	Π	P
	ϖ	2		
Rho	ρ	r	P	R
(Cursive)	ϱ	9		
Sigma	σ	s	Σ	S
(Final)	ς	v		
Tau	τ	t	T	T
Upsilon	υ	u	Υ	U
Phi	ϕ	f	Φ	F
(Cursive)	φ	4		
Chi	χ	c	X	C
Psi	ψ	y	Ψ	Y
Omega	ω	w	Ω	W

declaration subset for BK-1, ART-1, or SER-1:

```
<!ENTITY % aapmth1 PUBLIC "-//USA/AAP//DTD MATH-1
//EN"                    -- displayed formula -->
%aapmth1;
<!ENTITY % aapmth2 PUBLIC "-//USA/AAP//DTD MATH-2
//EN"                    -- in-line formula -->
%aapmth2;
```

TABLE B. German Characters

Lowercase		*Uppercase*	
Character	*Keyed*	*Character*	*Keyed*
𝔞	a	𝔄	A
𝔟	b	𝔅	B
𝔠	c	ℭ	C
𝔡	d	𝔇	D
𝔢	e	𝔈	E
𝔣	f	𝔉	F
𝔤	g	𝔊	G
𝔥	h	ℌ	H
𝔦	i	ℑ	I
𝔧	j	𝔍	J
𝔨	k	𝔎	K
𝔩	l	𝔏	L
𝔪	m	𝔐	M
𝔫	n	𝔑	N
𝔬	o	𝔒	O
𝔭	p	𝔓	P
𝔮	q	𝔔	Q
𝔯	r	ℜ	R
𝔰	s	𝔖	S
𝔱	t	𝔗	T
𝔲	u	𝔘	U
𝔳	v	𝔙	V
𝔴	w	𝔚	W
𝔵	x	𝔛	X
𝔶	y	𝔜	Y
𝔷	z	ℨ	Z

```
<!ENTITY % p.zz "(%p.e1;)|(%p.tbl;)|(%p.lst.d;)
                 |(%form1;)|(%form2;)"
              -- paragraph subelements -->
<!ENTITY % p.rf.ph "ntr|fnr|fgr|tbr|artr|apr|srr|rb|lar">
```

(The entity "p.em.ph" must be defined before invoking %aapmth1; and %aapmth2;.)

In addition, entity declarations for all entity references used in the document must be included. They may be incorporated individually if (1) there are only a few and/or (2) they are user defined entity references. Otherwise, they may be incorporated by invoking the appropriate public entity sets.

MATHEMATICAL FORMULAS AS STAND-ALONE DOCUMENTS

The AAP document type declarations for mathematical formulas were designed to be incorporated into one of the other AAP document types. If you choose to process mathematical formulas as stand-alone documents, however, the following modifications must be made to the document type declarations for mathematical formulas:

1. The document type declaration syntax must be used as follows:
   ```
   <!DOCTYPE math_doc [
   -- insert document type declaration subset as described
   below --
   ]>
   ```
2. Parameter entity declarations for the following parameter entity references must be created before invoking the declarations for mathematical formulas:
   - %p.em.ph; (emphasis)
   - %p.lst.d; (lists)
   - %p.rf.ph; (references to other elements)
3. Element declarations for the following elements must be created:
   - paragraph (<p>)
   - quotation, in-line (<q>)
   - any elements included in %p.em.ph; above
   - any elements included in %p.lst.d; above
   - any elements included in %p.rf.ph; above
4. Short reference declarations for any short references (excluding those already declared in the document type declaration subset for mathematical formulas) to be used in the document must be created.
5. General entity declarations must be incorporated into the document type declaration subset for any entity references used in the document. They may be included individually and/or incorporated by invoking the appropriate public entity sets.
6. The notation declaration and element declaration for the Cyrillic element (<cyr>) must be added.

APPENDIX B

FULL MARKUP OF MATHEMATICAL EXAMPLES

[1] <fd><la>1.1</ la>

[2] <fd><la loc = pre>1.1</ la>

[3] <fd><la loc = post>1.1</ la>

[4] <fd><la loc = pre>1.1.a</ la><la loc = post>1.1.b</ la>

[5] <fr><nu>1</ nu><de>2</ de></ fr>

[6] <fr><nu>2x</ nu><de>3xy</ de></ fr>

[7] <fr><nu>2x + 3y</ nu><de>12x−12y</ de></ fr>

[8] <fr align = r><nu>a + b + c</ nu><de>x + y</ de></ fr>

[9] <fr style = b><nu>1</ nu><de>1 + <fr><nu>1</ nu>
<de>y</ de></ fr></ de></ fr>

[10] <lim><op>∪</ op><ll>n = 1</ ll>
m</ ul>
<opd><fen lp = par>x<inf>n</ inf>∪y<inf>n</ inf>
<rp post = par></ fen></ opd></ lim>

[11] <sum><ll>i = 1</ ll>
i = 10</ ul><opd>x<inf>i</ inf>
</ opd></ sum>

[12] <rad><rcd>a + b</ rcd><rdx>3</ rdx></ rad>

[13] <fen lp = par>0, 1<rp post = par></ fen>

[14] <fen lp = sqb><fr><nu>A + 1</ nu><de>B</ de></ fr>
<rp post = sqb></ fen>

[15] <fen><fr><nu>A + 1</ nu><de>B</ de></ fr><rp></ fen>

[16] <fen style = b><fr><nu>A + 1</ nu><de>B</ de></ fr>
<rp style = b></ fen>

[17] <fen style = d><fr><nu>A + 1</ nu><de>B</ de></ fr>
<rp style = d></ fen>

[18] <fen style = n><fr><nu><inc>df</ inc></ nu><de><inc>dx</ inc>
</ de></ fr><fen lp = par>y<rp post = par></ rp>
</ fen><rp></ fen><inf>x = 0</ inf>

[19] <fen lp = ang><g>a</ g> + <g>b</ g><cp><fr><nu><inc><g>6</ g>
<g>a</ g></ inc></ nu><de><inc><g>6</ g>p</ inc></ de></ fr>
<rp post = ang></ fen>

[20] <fen lp = par><ar><arr><arc>1a</ arc><arc>1b</ arc>
<arc>1c</ arc></ arr><arr><arc>2a</ arc><arc>2b</ arc>
<arc>2c</ arc></ arr></ ar><rp post = par></ fen>

[21] <fen lp = par><ar><arr><arc>1</ arc></ arr>
<arr><arc></ arc><arc>1</ arc></ arr>
<arr><arc></ arc><arc></ arc><arc>1</ arc></ arr></ ar>
<rp post = par></ fen>

[22] <fen lp = par><ar>
<arr><arc>1</ arc></ arr>
<arr><arc></ arc><arc>1</ arc></ arr>
<arr><arc></ arc><arc></ arc><arc>1</ arc></ arr>
</ ar><rp post = par></ fen>

[23] <ar cs = "0 d F d A s" rs = "0 d F d A s">

[24] <ar ca = "2 d"><arr><arc>1</ arc><arc>2.1</ arc></ arr>
<arr><arc>2</ arc><arc>3.64</ arc></ arr></ ar>

[25] <arr><arc>492</ arc><arc>362</ arc><arc align
= l>5%</ arc></ arr>

[26] <stk><lyr>a</ lyr>
<lyr>ABCBA</ lyr>
<lyr>i<g>w</ g>t</ lyr></ stk>

[27] <stk><lyr>a</ lyr><lyr align = l>ABCBA</ lyr>
<lyr align = r>i<g>w</ g>t</ lyr></ stk>

[28] T<sup>1</ sup><inf>2</ inf>

[29] e<sup>x<sup>2</ sup></ sup>

[30] 2<sup>n<inf>1</ inf></ sup>

[31] E<inf><g>a</ g><sup>3</ sup></ inf>

[32] e<sup>a<sup>1</ sup><inf>2</ inf></ sup>

[33] T<inf>1</ inf><zw><sup>2</ sup><zw><inf>3</ inf>

[34] <fen lp = par><fr><nu>a</ nu><de>b</ de></ fr><rp post = par>
</ fen><sup loc = pre>1</ sup><inf loc = pre>2</ inf>
<sup>cd</ sup><inf>ab</ inf>

[35] <box>a + b</ box>

[36] <unl style = da>a + b</ unl>

[37] <ovl>a + b</ ovl>

[38] <box style = b>a + b</ box>

[39] <a><ac>x</ ac><ac>→</ ac></ a>

[40] <a><ac>xy</ ac><ac>→</ ac></ a>

[41] <a><ac><a><ac>x</ ac><ac>ˆ</ ac></ a></ ac>
<ac>→</ ac></ a>

[42] <a valign = d><ac><a><ac>abc</ ac><ac>ˆ</ ac></ a></ ac>
<ac>→</ ac></ a>

[43] <a><ac><a valign = d><ac>abc</ ac><ac>→</ ac></ a></ ac>
<ac>ˆ</ ac></ a>

[44] <a><ac><a><ac>x</ ac><ac>ˆ</ ac></ a></ ac>
<ac>→</ ac></ a>

[45] <rf>Tr</ rf>

```
[46] <inc>dx</inc>

[47] <inc><g>d</g>u&prime;</inc>

[48] <v>v<inf>x</inf></v> = <fen lp = par>1,0,0<rp post = par></rp>
     </fen>

[49] <v>B</v>&times;<v>V</v>

[50] <dy>A</dy>

[51] <dy>A</dy>

[52] <fi>R</fi><sup>2</sup>

[53] <fi>N</fi>

[54] <sum><ll>i = 0<tu>n = 1</ll><ul>i = 5<tu>n = 7</ul>
     <opd>2n<inf>i</inf></opd></sum>

[55] <fd><fl>y = <sum><ll></ll><ul></ul><opd>x<inf>i</inf></opd>
     </sum>,<hsp><phr>where</phr><hsp sp = 1>
     x<inf>i</inf> = r<sup>2</sup><inf>1</inf><hsp sp = 1>
     <phr>for</phr><hsp sp = .5>1&les;i&les;n</fl></fd>

[56] <ach atom = close>x</ach>

[57] <fd><fl>f<fen lp = par>x,y<rp post = par></fen> = 0
     <hsp sp = 1><phr>if and only if</phr>
     <hsp sp = 1>x = 0<hsp sp = .5><phr>
     or</phr><hsp sp = .5>y = 0</fl></fd>

[58] <fd><fl>f<fen lp = par>x,y<rp post = par></fen> = 0<phr>
     if and only if</phr>x = 0<phr>or</phr>y = 0</fl></fd>

[59] <fd><fl>S &prop;<rad><rcd><phr>word length,</phr></rcd>
     </rad><hsp><phr>force</phr> = <phr>mass</phr>
     &times;<phr>acceleration</phr></fl></fd>

[60] <fd><fl>S &prop;<rad><rcd><phr>word length,</rcd></rad>
     <phr>force</phr> = <phr>mass</phr>&times;
     <phr>acceleration</phr></fl></fd>

[61] <lar>1.1</lar>
```


APPENDIX C

ENTITY REFERENCES FOR USE IN MATHEMATICAL FORMULAS

The following entity references are defined for your use when creating mathematical formulas. They are divided into groups according to the public entity sets in which they occur.

General Special Graphic Characters (ISO 8879-1986 Numeric and Special Graphic)

Symbol	*Entity Reference*	*Description*
±	±	plus-or-minus sign – \pm
÷	÷	divide sign – \div
×	×	multiply sign – \times
*	*	asterisk – \ast
[	[	left square bracket – \lbrack
\	\	reverse solidus – \backslash
]	]	right square bracket – \rbrack
{	{	left curly bracket – \lbrace
|			vertical bar – \vert
}	}	right curly bracket – \rbrace
·	˙	middle dot – \centerdot
←	←	leftward arrow – \leftarrow
→	→	rightward arrow – \rightarrow
↑	↑	upward arrow – \uparrow
↓	↓	downward arrow – \downarrow

General Use Publishing Characters (ISO 8879-1986 Publishing)

Symbol	*Entity Reference*	*Description*
–	−	minus sign
†	†	dagger – \dagger
‡	‡	double dagger – \ddagger
✠	✠	maltese cross – \maltese

Symbol	Entity Reference	Description
♣	♣	club suit symbol – \clubsuit
♦	♦	diamond suit symbol – \diamondsuit
♥	♥	heart suit symbol – \heartsuit
♠	♠	spades suit symbol – \spadesuit
◊	◊	lozenge or total mark – \lozenge
△	▵	up triangle, open – \triangle
▽	▿	down triangle, open – \triangledown
▷	▹	r triangle, open – \triangleright
◁	◃	l triangle, open – \triangleleft
♭	♭	music flat – \flat
◆	⧫	lozenge, filled – \blacklozenge
•	•	dot bullet, filled – \bullet
■	▪	sq bullet, filled – \blacksquare
▲	▴	up tri, filled – \blacktriangle
▼	▾	dn tri, filled – \blacktriangledown
◄	◂	l tri, filled – \blacktriangleleft
►	▸	r tri, filled – \blacktriangleright
♮	♮	music natural – \natural
♯	♯	music sharp – \sharp

Technical Use Set 1 (ISO 8879-1986 General Technical)

Symbol	Entity Reference	Description
≡	≡	identical with – \equiv
∨	∨	logical or – \vee \lor
∧	∧	logical and – \wedge \land
∩	∩	intersection – \cap
∪	∪	union or logical sum – \cup
⊂	⊂	subset or is implied by – \subset
⊃	⊃	superset or implies – \supset
⇐	⇐	is implied by – \Leftarrow
⇒	⇒	implies – \Rightarrow
∴	∴	therefore – \therefore
∵	∵	because – \because
∈	∈	set membership – \in
∋	∋	contains – \ni
⊆	⊆	subset, equals – \subseteq
⊇	⊇	superset, equals – \supseteq
⇔	⇔	if and only if – \iff
¬	¬	logical not – \neg
∀	∀	for all – \forall
∃	∃	at least one exists – \exists
∫	∫	integral operator – \int
∮	∮	contour integral operator – \oint
∞	∞	infinity – \infty
~	∼	similar – \sim

Symbol	Entity Reference	Description
≈	≈	approximate – \approx
≃	≃	similar, equals – \simeq
≅	≅	congruent with – \cong
≦	≤	less-than-or-equal – \leq \le
≠	≠	not equal – \ne \neq
≧	≥	greater-than-or-equal – \geq \ge
∝	∝	is proportional to – \propto
ℵ	ℵ	aleph, Hebrew – \aleph
□	□	square – \square
∥	∥	parallel – \parallel
⊥	⊥	perpendicular – \bot
⊥	⊥	perpendicular – \perp
∢	∢	angle-spherical – \sphericalangle
∟	&ang90;	right (90 degree) angle
〈	⟨	left angle bracket – \langle
〉	⟩	right angle bracket – \rangle
′	′	prime or minute – \prime
″	″	double prime or second
∓	∓	minus-or-plus sign – \mp
‴	‴	triple prime
*	&dast;	downward asterisk

Additional Mathematical Symbols: Miscellaneous (ISO 8879-1986 Added Math Symbols: Ordinary)

Symbol	Entity Reference	Description
∠	∠	angle – \angle
∡	∡	angle-measured – \measuredangle
ב	ℶ	beth, Hebrew – \beth
‵	‵	reverse prime – \backprime
∁	∁	complement sign – \complement
ד	ℸ	daleth, Hebrew – \daleth
∅	∅	small o, slash – \emptyset
ג	ℷ	gimel, Hebrew – \gimel
∄	∄	negated exists – \nexists
ℏ	&plank;	Planck's constant – \hbar
℘	℘	Weierstrass p – \wp

Additional Mathematical Symbols: Binary and Large Operators (ISO 8879-1986 Added Math Symbols: Binary Operators)

Symbol	Entity Reference	Description
⨿	⨿	amalgamation or coproduct – \amalg
⩛	⌆	log and, dbl bar – \doublebarwedge
⊼	⌅	logical and, bar above – \barwedge

Symbol	*Entity Reference*	*Description*
⋒	⋒	dbl intersection – \Cap
⋓	⋓	dbl union – \Cup
⋎	⋎	curly logical or – \curlyvee
⋏	⋏	curly logical and – \curlywedge
◇	⋄	open diamond – \diamond
⋇	⋇	division on times – \divideontimes
⊺	⊺	intercal – \intercal
⋋	⋋	\leftthreetimes
⋉	⋉	times sign, left closed
⊟	⊟	minus sign in box – \boxminus
⊛	⊛	asterisk in circle – \circledast
⊚	ô	open dot in circle – \circledcirc
⊝	⊝	hyphen in circle – \circleddash
⊙	⊙	middle dot in circle – \odot
⊖	⊖	minus sign in circle – \ominus
⊕	⊕	plus sign in circle – \oplus
⊘	⊘	solidus in circle – \oslash
⊗	⊗	multiply sign in circle – \otimes
⊞	⊞	plus sign in box – \boxplus
∔	∔	plus sign, dot above – \dotplus
⋌	⋌	\rightthreetimes
⋊	⋊	times sign, right closed – \rtimes
⊡	⊡	small dot in box – \dotsquare \boxdot
∖	∖	reverse solidus – \setminus
⊓	⊓	square intersection – \sqcap
⊔	⊔	square union – \sqcup
★	⋆	small star, filled – \star
⊠	⊠	multiply sign in box – \boxtimes
⊤	⊤	inverted perpendicular – \top
⊎	⊎	plus sign in union – \uplus
≀	≀	wreath product – \wr
▽	▽	big dn tri, open – \bigtriangledown
△	△	big up tri, open – \bigtriangleup

Additional Mathematical Symbols: Relations (ISO 8879-1986 Added Math Symbols: Relations)

Symbol	*Entity Reference*	*Description*
≊	≊	approximate, equals – \approxeq
≍	≈	asymptotically equal to – \asymp
≌	≌	reverse congruent – \backcong
⋈	⋈	\bowtie
∽	∽	reverse similar – \backsim
⋍	⋍	reverse similar, eq – \backsimeq
≎	≎	bumpy equals – \Bumpeq

Symbol	*Entity Reference*	*Description*
≏	≏	bumpy equals, equals – \bumpeq
≗	&circe;	circle, equals – \circeq
≔	≔	colon, equals – \coloneq
≼	⋞	curly eq, precedes – \curlyeqprec
≽	⋟	curly eq, succeeds – \curlyeqsucc
⋞	&cupre;	curly precedes, eq – \curlypreceq
⊣	⊣	dash, vertical – \dashv
≖	≖	circle on equals sign – \eqcirc
≕	≕	equals, colon – \eqcolon
≑	≑	eq, even dots – \Doteq
≒	ė	equals, dot above – \doteq
≒	≒	eq, falling dots – \fallingdotseq
⩾	⪖	equal-or-gtr, slanted – \eqslantgtr
⩽	⪕	eq-or-less, slanted – \eqslantless
≓	≓	eq, rising dots – \risingdotseq
⋔	⋔	pitchfork – \pitchfork
⌢	⌢	down curve – \frown
⪆	⪆	greater, approximate – \gtrapprox
⋗	ġ	greater than, with dot – \gtrdot
≧	≧	greater, double equals – \geqq
⋛	⋛	greater, equals, less – \gtreqless
⪌	⪌	gt, dbl equals, less – \gtreqqless
⩾	⩾	gt-or-equal, slanted – \geqslant
⋙	⋙	triple gtr-than – \ggg
≷	≷	greater, less – \gtrless
≳	≳	greater, similar – \gtrsim
≫	≫	dbl greater-than sign – \gg
⪅	⪅	less, approximate – \lessapprox
⋖	&ldot;	less than, with dot – \lessdot
≦	≦	less, double equals – \leqq
⪋	⪋	less, dbl eq, greater – \lesseqqgtr
⋚	⋚	less, eq, greater – \lesseqgtr
⩽	⩽	less-than-or-eq, slant – \leqslant
≶	≶	less, greater – \lessgtr
⋘	⋘	triple less-than – \lll
≲	≲	less, similar – \lesssim
≪	≪	double less-than sign – \ll
⊴	⊴	left triangle, eq – \trianglelefteq
∣	∣	\mid
⊧	⊧	\models
≺	≺	precedes – \prec
⪷	⪷	precedes, approximate – \precapprox
≼	⪯	precedes, equals – \preceq
≾	≾	precedes, similar – \precsim
⊵	⊵	right tri, eq – \trianglerighteq
⨿	&samalg;	small amalg – \smallamalg
≻	≻	succeeds – \succ
⪸	⪸	succeeds, approximate – \succapprox
≽	≽	succeeds, curly eq – \succcurlyeq

Symbol	*Entity Reference*	*Description*
≽	⪰	succeeds, equals – \succeq
≿	≿	succeeds, similar – \succsim
⌢	⌢	small down curve – \smallfrown
∣	∣	\shortmid
⌣	⌣	up curve – \smile
∥	∥	short parallel – \shortparallel
⊏	⊏	square subset – \sqsubset
⊑	⊑	square subset, equals – \sqsubseteq
⊐	⊐	square superset – \sqsupset
⊒	⊒	square superset, eq – \sqsupseteq
⌣	⌣	small up curve – \smallsmile
⋐	⋐	double subset – \Subset
⊆	⫅	subset, dbl equals – \subseteqq
⋑	⋑	dbl superset – \Supset
⊇	⫆	superset, dbl equals – \supseteqq
≜	≜	triangle, equals – \triangleq
≬	≬	between – \between
⊢	⊢	vertical, dash – \vdash
⊩	⊩	dbl vertical, dash – \Vdash
⩚	⊻	logical or, bar below – \veebar
⊪	⊪	triple vertical, dash – \Vvdash

Additional Mathematical Symbols: Negated Relations (ISO 8879-1986 Added Math Symbols: Negated Relations)

Symbol	*Entity Reference*	*Description*
⪊	⪊	greater, not approximate – \gnapprox
⪈	⪈	greater, not equals – \gneq
≩	≩	greater, not dbl equals – \gneqq
⋧	⋧	greater, not similar – \gnsim
⪉	⪉	less, not approximate – \lnapprox
⪇	≨	less, not double equals – \lneqq
⪇	⪇	less, not equals – \lneq
⋦	⋦	less, not similar – \lnsim
≉	≉	not approximate – \napprox
≇	≇	not congruent with – \ncong
≢	≢	not identical with – \nequiv
≱	≧̸	not greater, dbl equals – \ngeqq
≱	≱	not greater-than-or-equal – \ngeq
⩾̸	⩾̸	not gt-or-eq, slanted – \ngeqslant
≯	≯	not greater-than – \ngtr
≰	≰	not less-than-or-equal – \nleq
≦̸	≦̸	not less, dbl equals – \nleqq
⩽̸	⩽̸	not less-or-eq, slant – \nleqslant
≮	≮	not less-than – \nless
⋪	⋪	not left triangle – \ntriangleleft
⋬	⋬	not l tri, eq – \ntrianglelefteq
∤	∤	\nmid

Symbol	*Entity Reference*	*Description*
∦	∦	not parallel – \nparallel
⊀	⊀	not precedes – \nprec
⪯̸	⪯̸	not precedes, equals – \npreceq
⋫	⋫	not rt triangle – \ntriangleright
⋭	⋭	not r tri, eq – \ntrianglerighteq
⊁	⊁	not succeeds – \nsucc
⪰̸	⪰̸	not succeeds, equals – \nsucceq
≁	≁	not similar – \nsim
≄	≄	not similar, equals – \nsimeq
∤	∤	\nshortmid
∦	∦	not short par – \nshortparallel
⊄	⊄	not subset – \nsubset
⊈	⊈	not subset, equals – \nsubseteq
⫅̸	⫅̸	not subset, dbl eq – \nsubseteqq
⊅	⊅	not superset – \nsupset
⊉	⫆̸	not superset, dbl eq – \nsupseteqq
⊉	⊉	not superset, equals – \nsupseteq
⊬	⊬	not vertical, dash – \nvdash
⊭	⊭	not vertical, dbl dash – \nvDash
⊯	⊯	not dbl vert, dbl dash – \nVDash
⊮	⊮	not dbl vertical, dash – \nVdash
⪹	⪹	precedes, not approx – \precnapprox
⪵	⪵	precedes, not dbl eq – \precneqq
⋨	⋨	precedes, not similar – \precnsim
⪺	⪺	succeeds, not approx – \succnapprox
⪶	⪶	succeeds, not dbl eq – \succneqq
⋩	⋩	succeeds, not similar – \succnsim
⫋	⊊	subset, not equals – \subsetneq
⫋	⫋	subset, not dbl eq – \subsetneqq
⫌	⊋	superset, not equals – \supsetneq
⫌	⫌	superset, not dbl eq – \supsetneqq

Additional Mathematical Symbols: Arrow Relations (ISO 8879-1986 Added Math Symbols: Arrow Relations)

Symbol	*Entity Reference*	*Description*
↶	↶	left curved arrow – \curvearrowleft
↷	↷	rt curved arrow – \curvearrowright
⇓	⇓	down dbl arrow – \Downarrow
⇊	&darr2;	two down arrows – \downdownarrows
⇃	⇃	dn harpoon-left – \downleftharpoon
⇂	⇂	down harpoon-rt – \downrightharpoon
↙	&dlarr;	downward l arrow – \swarrow
↘	&drarr;	downward rt arrow – \searrow
⇔	⇔	l & r dbl arrow – \Leftrightarrow
↔	↔	l & r arrow – \leftrightarrow
↭	↭	l & r arr-wavy – \leftrightsquigarrow
↞	↞	\twoheadleftarrow

Symbol	Entity Reference	Description
⇚	⇚	left triple arrow – \Lleftarrow
⇇	&larr2;	two left arrows – \leftleftarrows
↩	↩	left arrow-hooked – \hookleftarrow
↫	↫	left arrow-looped – \looparrowleft
↢	↢	left arrow-tailed – \leftarrowtail
↽	↽	l harpoon-down – \leftharpoondown
↼	↼	left harpoon-up – \leftharpoonup
⇆	&lrarr2;	l arr over r arr – \leftrightarrows
⇋	&lrhar2;	l harp over r – \leftrightharpoons
↰	↰	\Lsh
↦	↦	\mapsto
⟼	⊸	\multimap
↗	↗	NE pointing arrow – \nearrow
⇍	⇍	not implied by – \nLeftarrow
↚	↚	not left arrow – \nleftarrow
⇎	⇎	not l & r dbl arr – \nLeftrightarrow
↮	↮	not l & r arrow – \nleftrightarrow
↛	↛	not right arrow – \nrightarrow
⇏	⇏	not implies – \nRightarrow
↖	↖	NW pointing arrow – \nwarrow
↺	↺	l arr in circle – \circlearrowleft
↻	↻	r arr in circle – \circlearrowright
⇻	↠	\twoheadrightarrow
⇛	⇛	right triple arrow – \Rrightarrow
⇉	&rarr2;	two rt arrows – \rightrightarrows
↪	↪	rt arrow-hooked – \hookrightarrow
↬	↬	rt arrow-looped – \looparrowright
↣	↣	rt arrow-tailed – \rightarrowtail
⇝	↝	rt arrow-wavy – \squigarrowright
⇁	⇁	rt harpoon-down – \rightharpoondown
⇀	⇀	rt harpoon-up – \rightharpoonup
⇄	&rlarr2;	r arr over l arr – \rightleftarrows
⇌	&rlhar2;	r harp over l – \rightleftharpoons
↱	↱	\Rsh
⇑	⇑	up dbl arrow – \Uparrow
⇈	&uarr2;	two up arrows – \upuparrows
⇕	⇕	up & down dbl arrow – \Updownarrow
↕	↕	up & down arrow – \updownarrow
↿	↿	up harpoon-left – \upleftharpoon
↾	↾	up harpoon-right – \uprightharpoon
⇔	⟺	long l & r dbl arr – \Longleftrightarrow

Additional Mathematical Symbols: Closing and Opening Delimiters (ISO 8879-1986 Added Math Symbols: Delimiters)

Symbol	Entity Reference	Description
)>	⦔	right paren, gt – \rightparengtr
(>	&lpargt;	left paren, gt – \leftparengtr

APPENDIX D

COMPREHENSIVE LIST OF MATHEMATICAL CONSTRUCTS WITH TAGS

Array	<ar>
Array cell	<arc>
Array row	<arr>
Atom change	<ach>
Boxed element	<box>
Dyadic	<dy>
Accented element	<a>
Accented element component	<ac>
Fence element	<fen>
Post, center	<cp>
Post, right	<rp>
Field	<fi>
Formula, displayed	<fd>
Formula line	<fl>
Label	<la>
Formula, in-line	<f>
Fraction	<fr>
Denominator	<de>
Numerator	<nu>
Horizontal mark	<hmk>
Horizontal mark, reference to	<hmkr>
Horizontal space	<hsp>
Increment	<inc>
Inferior	<inf>
Limits	<lim>
Limit, lower	<ll>
Limit, upper	
Operand	<opd>
Operator	<op>
Special limits	
Integral limits	<in>
Product limits	<pr>
Summation limits	<sum>
Limit, lower	<ll>
Limit, upper	
Operand	<opd>
Overlined element	<ovl>
Phrase (nonmath text)	<phr>
Label, reference to	<lar>
Radical	<rad>
Index for radical	<rdx>
Radicand	<rcd>
Roman function	<rf>
Stack	<stk>
Layer	<lyr>
Superior	<sup>
Turn line	<tu>
Underlined element	<unl>
Vector	<v>
Vertical mark	<vmk>
Vertical mark, reference to	<vmkr>
Vertical space	<vsp>

Type Style Tags

Bold	<b>
Bold German	<bge>
Bold Greek	<bg>
Bold italic	<bi>
Bold italic sans serif	<bisf>
Bold sans serif	<bsf>
Bold script	<bsc>
Cyrillic	<cyr>
German	<ge>
Greek	<g>
Italic	<it>
Italic sans serif	<isf>
Monospace (typewriter)	<ty>
Openface	<op>
Roman	<rm>
Sans serif	<ssf>
Script (fancy)	<sc>
Small caps	<scp>

APPENDIX E

DESCRIPTION OF SPECIAL SYMBOLS IN TABLE 1

&ab;	down left harpoon
∾	down right harpoon
&ad;	down arrow
&ae;	down double arrow (Downarrow)
⁡	left harpoon over right (leftrightharpoons)
&ag;	right harpoon over left (rightleftharpoons)
&ah;	left arrow over right arrow (leftrightarrows)
&ai;	right arrow over left arrow (rightleftarrows)
&aj;	hook left arrow
&ak;	left harpoon up
&al;	left arrow
&am;	is implied by (Leftarrow)
&an;	left right squig arrow
&ao;	squig arrow right
≈	hook right arrow
&aq;	right harpoon up
&ar;	right arrow
&as;	implies (Right arrow)
&at;	maps to
&au;	up arrow
&av;	up double arrow (Uparrow)
&aw;	up left harpoon
&ax;	up right harpoon
&ay;	right triple arrow (Rrightarrow)
&az;	left triple arrow (Lleftarrow)
&a0;	up and down arrow (updownarrow)
&a1;	up and down double arrow (Updownarrow)
&a2;	left right arrow
&a3;	left right double arrow (Leftrightarrow)
&a4;	NW pointing arrow (nwarrow)
&a5;	SE pointing arrow (searrow)
&a6;	NE pointing arrow (nearrow)
&a7;	SW pointing arrow (swarrow)

&a8;	two up arrows (upuparrows)
&a9;	two down arrows (downdownarrows)
&a10;	two right arrows (rightrightarrows)
&a11;	two left arrows (leftleftarrows)
&a12;	left arrow in circle (circlearrowleft)
&a13;	right arrow in circle (circlearrowright)
&a14;	curve arrow left
&a15;	curve arrow right
&a16;	left harpoon down
&a17;	right harpoon down
&a21;	right arrow tail
&a22;	left arrow tail
&a23;	if and only if (Longleftrightarrow)
&a24;	two head left arrow
&a25;	two head right arrow
&a26;	loop arrow left
&a27;	loop arrow right
&a28;	two headed mapping
&a29;	Lsh
&a30;	Rsh
&va;	subset, double equals (subseteqq)
&vb;	subset, equals (subseteq)
&vc;	subset
&vd;	superset, double equals (supseteqq)
&ve;	superset, equals (supseteq)
&vf;	superset (supset)
&vg;	triple greater than (ggg)
&vh;	circle on equals sign (eqcirc)
&vi;	is proportional to (propto)
&vj;	double vertical, dash (Vdash)
&vk;	double vertical, double dash (VDash)
&vl;	triple less-than (lll)
&vm;	triple vertical, dash (Vvdash)
&vn;	less-than-or-equals (leq)
&vo;	greater-than-or-equals (geq)
&vp;	perpendicular (perp)
&vq;	parallel
&vr;	triangle, equals (triangleq)
&vs;	equals, even dots (Doteq)
&vt;	bumpy equals, equals (bumpeq)
&vu;	bumpy equals (Bumpeq)
&vv;	mid
&vw;	less, double equals (leqq)
&vx;	greater, double equals (geqq)
&vy;	less, similar (lesssim)
&vz;	greater, similar (gtrsim)
&v0;	reverse congruent (backcong)
&v1;	similar, equals (simeq)
&v2;	approximately, equals (approxeq)
&v3;	between
&v4;	less, approximate (lessapprox)

&v5;	greater, approximate (gtrapprox)
&v6;	left elongated triangle
&v7;	right elongated triangle
&v8;	less, double equals, greater (lesseqqgtr)
&v9;	greater, double equals, less (gtreqqless)
&v10;	precedes, similar (precsim)
&v11;	succeeds, similar (succsim)
&v12;	precedes, approximate (precapprox)
&v13;	succeeds, approximate (succapprox)
&v14;	precedes (prec)
&v15;	succeeds (succ)
&v16;	curly precedes, equals (curlypreceq)
&v18;	congruent with (cong)
&v19;	square subset (sqsubset)
&v20;	square superset (sqsupset)
&v21;	square subset, equals (sqsubseteq)
&v22;	square superset, equals (sqsupseteq)
&v23;	reverse similar, equals (backsimeq)
&v24;	curly equals, precedes (curlyeqprec)
&v25;	curly equals, succeeds (curlyeqsucc)
&v26;	vertical, dash (vdash)
&v27;	models
&v28;	asymptotically equal to (asymp)
&v29;	succeeds, curly equals (succcurlyeq)
&v30;	triple tilde
&na;	subset, not double equals (subsetneqq)
&nb;	subset, not equals (subsetneq)
&nc;	not subset (nsubset)
&nd;	superset, not double equals (supsetneqq)
≠	superset, not equals (supsetneq)
&nf;	not superset (nsupset)
&ng;	not greater-than (ngtr)
&nh;	not precedes (nprec)
∋	not succeeds (nsucc)
&nj;	not precedes, equals (npreceq)
&nk;	not succeeds, equals (nsucceq)
&nl;	not left arrow (nleftarrow)
&nm;	not implied by (nLeftarrow)
&nn;	less, not equals (lneq)
&no;	greater, not equals (gneq)
&np;	negated exists (nexists)
&nq;	not parallel (nparallel)
&nr;	not right arrow (nrightarrow)
&ns;	not implies (nRightarrow)
&nt;	not similar (nsim)
ν	not approximate (napprox)
&nv;	not mid (nmid)
&nw;	less, not double equals (lneqq)
&nx;	greater, not double equals (gneqq)
&ny;	less, not similar (lnsim)
&nz;	greater, not similar (gnsim)

&n0;	not double vertical, dash (nVdash)
&n1;	not similar, equals (nsimeq)
&n2;	not double vertical, double dash (nVDash)
&n3;	not left and right double arrow (nLeftrightarrow)
&n4;	less, not approximate (lnapprox)
&n5;	greater, not approximate (gnapprox)
&n6;	does not contain as a member (notin)
&n7;	is not a member of (notni)
&n8;	not less-or-equal, slanted (nleqslant)
&n9;	not greater-or-equals, slanted (ngeqslant)
&n10;	precedes, not similar (precnsim)
&n11;	succeeds, not similar (succnsim)
&n12;	precedes, not approximate (precnapprox)
&n13;	succeeds, not approximate (succnapprox)
&n14;	precedes, not double equals (precneqq)
&n15;	succeeds, not double equals (succneqq)
&n16;	not subset, equals (nsubseteq)
&n17;	not superset, equals (nsupseteq)
&n18;	not congruent with (ncong)
&n19;	not greater-than-or-equals (ngeq)
&n20;	not less-than-or-equal (nleq)
&n21;	not less-than (nless)
&n22;	not left triangle (ntriangleleft)
&n23;	not right triangle (ntriangleright)
&n24;	not subset, double equals (nsubseteqq)
&n25;	not superset, double equals (nsupseteqq)
&n26;	not vertical, dash (nvdash)
&n27;	not vertical, double dash (nvDash)
&n28;	not identical with (nequiv)
&n30;	not triple tilde
&ja;	for all
&jb;	at least one exists (exists)
&jc;	left parenthesis, greater-than (leftparengtr)
&jd;	right parenthesis, greater-than (rightparengtr)
&je;	infinity (infty)
&jf;	less-than-or-equal, slanted (leqslant)
&jg;	greater-than
&jh;	Planck's constant (hbar)
&ji;	integral operator
&jj;	precedes, equals (preceq)
&jk;	succeeds, equals (succeq)
&jl;	less-than
&jm;	minus-or-plus sign (mp)
&jn;	not equal (neq)
&jo;	greater-or-equal, slanted (geqslant)
&jp;	short parallel
&jq;	curly logical or (curlyvee)
&jr;	multimap
&jt;	similar (sim)
&jv;	short mid
&jx;	multiply sign (times)

&jy;	division on times
&j0;	degrees
&j2;	not left and right arrow (nleftrightarrow)
&j4;	double dagger (ddagger)
&j5;	dagger
&j6;	reverse prime (backprime)
&j7;	prime
&j8;	double prime
&j9;	triple prime
&j10;	middle dot (centereddot)
&j12;	maltese cross
&j13;	divide sign
&j14;	up curve (smile)
&j15;	down curve (frown)
&j16;	per mille
&j17;	equals, falling dots (fallingdotseq)
&j19;	left angle bracket (langle)
&j20;	right angle bracket (rangle)
&j26;	plus-or-minus sign (pm)
&j28;	identical with (equiv)
&j31;	downward asterisk (lower ast)
&ka;	curly logical and (curlywedge)
&kb;	crosshairs
&kc;	double subset (Subset)
&kd;	triangle conjunction
&ke;	circle vertical
&kf;	double superset (Supset)
&kg;	greater-than, with dot (gtrdot)
&ki;	cauchy integral
&kj;	pitchfork
&kl;	less-than, with dot (lessdot)
&km;	times sign, left closed (ltimes)
&kn;	times sign, right closed (rtimes)
&ko;	bowtie
&kp;	not short parallel
&kq;	double logical or (double vee)
&kr;	double logical and (double wedge)
&ks;	log and, double bar (doublebarwedge)
&kt;	intercal
&ku;	double union (Cup)
&kv;	not short mid
&kw;	double intersection (Cap)
&kx;	right three times
&ky;	left three times
&k0;	pentagon
&k1;	hexagon
&k2;	big triangle down
&k3;	big triangle up
&k5;	small amalgamation
&k6;	left triangle, equals (trianglelefteq)
&k7;	right triangle, equals (trianglerighteq)

&k8;	not left triangle, equals (ntrianglelefteq)
&k9;	not right triangle, equals (ntrianglerighteq)
&k10;	circle, equals (circeq)
&k11;	equals, dot above (doteq)
&k12;	equals, colon (eqcolon)
&k13;	colon, equals (coloneq)
&k14;	small up curve (smallsmile)
&k15;	small down curve (smallfrown)
&k16;	per 10,000
&k17;	equals, rising dots (risingdotseq)
&k18;	minus with dot beneath
&k19;	not greater, double equals (ngeqq)
&k20;	not less, double equals (nleqq)
&k21;	open star
&k22;	short triple bond
&k24;	left fishhook
&k25;	right fishhook
&k26;	plus with dot beneath
&k27;	double perpendicular
&k28;	short double bond
&k30;	slash, solidus
&k31;	centered asterisk
&la;	aleph, Hebrew
&lb;	beth, Hebrew
&ld;	daleth, Hebrew
≤	contains (ni)
&lf;	equal-or-less, slanted (eqslantless)
≶	double greater-than sign (gg)
&lh;	gimel, Hebrew
&li;	contour integral operator (oint)
&lj;	surface integral
&lk;	volume integral
≪	double less-than sign (ll)
&lm;	music flat (flat)
&ln;	music sharp (sharp)
&lo;	equal-or-greater, slanted (eqslantgtr)
&lp;	bottom (bot)
&ls;	square bullet, filled (blacksquare)
<	reverse similar (backsim)
&lu;	plus sign in union (uplus)
&lx;	multiply sign in box (boxtimes)
&lz;	small dot in box (dotsquare)
&l0;	dot bullet, filled (bullet)
&l1;	diamondsuit symbol
&l2;	heartsuit symbol
&l3;	convolution (circ)
&l4;	right (90 degree) angle
&l5;	spadesuit symbol
&l6;	clubsuit symbol
&l7;	amalgamation or coproduct (amalg)
&l8;	less, equals, greater (lesseqgtr)

&l9;	greater, equals, less (gtreqless)
&l10;	up triangle, filled (blacktriangle)
&l11;	down triangle, filled (blacktriangledown)
&l12;	left triangle, filled (blacktriangleleft)
&l13;	right triangle, filled (blacktriangleright)
&l14;	square union (sqcup)
&l15;	square intersection (sqcap)
&l18;	minus sign in box (boxminus)
&l22;	triple bond
&l25;	plus sign, dot above (dotplus)
&l26;	plus sign in box (boxplus)
&l28;	double bond
&l30;	set minus
&l31;	asterisk in circle (circledast)
&ma;	minus sign
&mb;	Mercury
&mc;	complement sign
&me;	set membership (in)
&mf;	Venus, female
&mg;	Jupiter
&mh;	Saturn
&mi;	CCW circuital integral
&mj;	CW circuital integral
&mk;	music natural
&mm;	Mars, male
&mo;	small o, slash (emptyset)
∓	Weierstrass p (wp)
&mq;	because
&mr;	therefore
&ms;	square
&mt;	top
μ	union; logical sum (cup)
&mv;	logical or (vee)
&mw;	intersection (cap)
&mx;	multiply sign in circle (otimes)
&my;	logical and (wedge)
&mz;	middle dot in circle (odot)
&m0;	big circle (bigcirc)
&m1;	total mark (lozenge)
&m2;	open diamond (diamond)
&m3;	Neptune
&m4;	angle
&m5;	angle-measured
&m6;	angle-spherical
&m7;	triple dot colon
&m8;	less, greater (lessgtr)
&m9;	greater, less (gtrless)
&m10;	up triangle open (triangle)
&m11;	down triangle open (triangledown)
&m12;	left triangle open (triangleleft)
&m13;	right triangle open (triangleright)

&m14;	logical or, bar below (veebar)
&m15;	logical and, bar above (barwedge)
&m16;	open dot in circle (circledcirc)
&m17;	hyphen in circle (circleddash)
&m18;	minus sign in circle (ominus)
&m19;	solidus in circle (oslash)
&m21;	small star, filled (star)
&m22;	wreath product (wr)
&m26;	plus sign in circle (oplus)
&m28;	approximate (approx)
≺	1/2-black square
&ps;	1/2-black square
&pt;	1/2-black square
&pu;	1/2-black square
&pv;	1/2-black square
&py;	approximate, less-than (approxlessthan)
&pz;	approximate, greater-than (approxgtrthan)
&p0;	1/2-black circle
&p1;	lozenge, filled (blacklozenge)
&p18;	logical not (neg)
&p26;	dash, vertical (dashv)
&p27;	NE–SW (45) stroke
&p30;	NW–SE (45) stroke

MATH COMMITTEE

Dr. James Roesser
Science Typographers Inc.
15 Industrial Boulevard
Medford, NY 11763

Dr. Bruce Foreman
American Institute of Physics
335 East 45th Street
New York, NY 10017

Mr. Ronald Whitney
American Mathematical Society
P.O. Box 6248
201 Charles Street
Providence, RI 02940

Mr. Barry C. W. Doherty
ATEX
15 Wiggins Avenue
Mail Stop 94/32W
Bedford, MA 01730

Mr. Joseph C. Fineman
239 Clinton Road
Brookline, MA 02146

Mr. Lawrence A. Beck
Grumman Data Systems
R & D—MS D12-237
1000 Woodbury Road
Woodbury, NY 11797

Mr. Lawrence A. Buckland
Inforonics
550 Newton
Littleton, MA 01460

Mr. Jordan Glogau
Graphic Technology Center
Suite 220B
9 West 57th Street
New York, NY 10019

Mr. George Alexander
J. B. Lippincott Co.
East Washington Square
Philadelphia, PA 19105

Mr. Lee Wingenroth
Science Press
300 West Chestnut Street
Ephrata, PA 17522

Mr. Eli Wylen
Texet Corporation
37 Broadway
Arlington, MA 02174

Ms. Mary E. Jones
John Wiley & Sons Inc.
600 3rd Avenue
10th Floor
New York, NY 10016

Mr. William W. Tunnicliffe
39 Central Street
Winchester, MA 01890

Mr. James D. Mason
Oak Ridge National Laboratory
Info. Div./Tech. Pub. Dept.
P.O. Box X
Oak Ridge, TN 37830

Mr. Marion Elledge
Graphic Communications Association
1730 North Lynn Street
Suite 604
Arlington, VA 22209

Mr. Robert F. Tipton
Science Press
300 West Chestnut Street
Ephrata, PA 17522

Mr. Peter Strupp
Plenum Publishing Corporation
233 Spring Street
New York, NY 10013

Mr. Herb Wasserberg
Bitstream Inc.
Anthenaeum House
215 First Street
Cambridge, MA 02142

Mr. Calvin Cox
Maryland Composition
6711 Dover Road
Glen Burnie, MD 21061

Mr. Richard Scheiding
Progressive Typographers
Box 278
Emigsville, PA 17318

Mr. Richard Deardorf
York Graphic Services Inc.
3600 W. Market Street
York, PA 17404

Mr. Frank Grohowski
Penta Systems International
1511 Guilford Avenue
Baltimore, MD 21202

Mr. Walter Kaufman-Buehler
Springer Verlag
175 Fifth Avenue
New York, NY 10010

IBM30123

Z286/E43A76/MARKUP.OF.MATHEMATICAL.FORMULAS

McPHERSON LIBRARY

University of Victoria

UVic-M
3 2775 009 154 20 3
Markup of mathematical formulas :
Z 286 E43A76

UNIVERSITY OF VICTORIA
Library
VICTORIA, B.C.

DUE DATE

MAR 0 1 1996